COMBINING STATISTICAL SIMILARITY MEASURES FOR AUTOMATIC INDUCTION OF SEMANTIC CLASSES

By

Apostolos E. Pangos

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
AT
TECHNICAL UNIVERSITY OF CRETE
CHANIA, GREECE
OCTOBER 2005

TECHNICAL UNIVERSITY OF CRETE
DEPARTMENT OF
ELECTRONICS AND COMPUTER ENGINEERING

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Combining Statistical Similarity Measures for Automatic Induction of Semantic Classes**" by **Apostolos E. Pangos** in partial fulfillment of the requirements for the degree of **Master of Science**.

Dated: October 2005

Supervisor: _____
Assoc. Prof. Alexandros Potamianos

Readers: _____
Prof. Vasilis Digalakis

_____
Prof. Stavros Christodoulakis

TECHNICAL UNIVERSITY OF CRETE

Date: **October 2005**

Author: **Apostolos E. Pangos**

Title: **Combining Statistical Similarity Measures for Automatic Induction of Semantic Classes**

Department: **Electronics and Computer Engineering**

Degree: **M.Sc.** Convocation: **October** Year: **2005**

Signature of Author

To Maria, Dimitris and Marina.

Table of Contents

List of Figures

Abstract

In this work, an unsupervised semantic class induction algorithm is proposed that is based on the principle that similarity of context implies similarity of meaning. Two semantic similarity metrics that are variations of the Vector Product distance are used in order to measure the semantic distance between words and to automatically generate semantic classes. The first metric computes "wide-context" similarity between words using a "bag-of-words" model, while the second metric computes "narrow-context" similarity using a bigram language model.

Additionally, a hybrid metric that is defined as the weighted linear combination of the wide and narrow-context metrics is also proposed and evaluated. The motivation of this approach is to improve effectiveness by combining evidence. Our work reinforces the idea that selecting the "best" algorithm is not necessarily the ideal choice, since potentially valuable information may be wasted by discarding the results of a less successful classifier. To calculate the values of the weights for each metric an algorithm is proposed that measures the quality of each metric. This algorithm is based on the inter-class distance.

To cluster words into semantic classes an iterative clustering algorithm is used. The semantic metrics are evaluated on two corpora: a semantically heterogeneous web news domain (HR-Net) and an application-specific travel reservation corpus (ATIS). For the hybrid metric, semantic class member precision of 90% is achieved at 24% recall for the HR-Net task and precision of 88% is achieved at 70% recall for the ATIS task. In general, "wide-context" similarity metric performs better in the heterogeneous corpus while "narrow-context" similarity metric in the domain specific corpus. The hybrid metric performs better in both cases.

Acknowledgements

I would like to express my sincere gratitude to my supervisor professor Alexandros Potamianos for his invaluable help, advice, guidance and encouragement throughout the development of this work. I would also like to thank professor Vassilis Digalakis for his useful recommendations. Also, it was my privilege to participate during the first two years of my studies in Chania, at the M.U.S.I.C. laboratory, headed by professor Stavros Christodoulakis.

All my gratitude to Elias Iosif for the help and advice that he offered me with generosity and for the countless hours that he spent for me. Many thanks to Christos Vosnidis for his help and support and to Eduardo Sanchez-Soto for reading and commenting my thesis.

I could not forget to thank my family, especially my parents, who supported me throughout my studies by all means. Special thanks to my fellow-traveller Panagiotis Karageorgakis, Christos Vosnidis, Lilian Saridaki and Kleanthis Mokios for their friendship. Finally, the claim that there exists no muse of understanding I know to be false, her name is Stavroula.

Chania, Crete — Apostolis Pangos
October 10, 2005

Chapter 1

Thesis Overview

Many applications dealing with textual information require classification of words into semantic classes including spoken dialogue systems, language modelling, speech understanding and machine translation applications. For example, in a typical human−machine spoken dialogue interaction over the telephone network, the telephone call is picked up by a telephony server, which passes the audio stream to a speech recognizer, residing on a remote computer. The automatic speech recognizer (ASR) transcribes the audio stream into a string of text, which is then passed on to a computer dialogue agent. The dialogue agent must extract the information content of this text string. This requires teaching the dialogue manager what semantic concepts to expect, and how to extract them from the persons utterance [21]. Text mining systems often convert text into a set of features, many of which are defined in terms of semantic classes [12]. In information extraction and question answering, many of the pattern matching rules make use of semantic classes such as management positions, expenditures, art work, etc. The subject of our work is the extraction of semantic concepts from text using an automatic procedure.

Manual construction of semantic classes is a time consuming task and often

requires expert knowledge: an inexperienced developer may omit important components for each semantic concept. It is a daunting and expensive task, which forms a major bottleneck in the development of language understanding systems. The difficulty arises because a new domain (or, task) for an application is frequently poorly understood because there may be very little a priori knowledge about grammar structure or the semantic meanings of words. A comprehensive description of the semantics and grammar for such tasks must then be defined manually. The set of semantic classes, where each semantic class is a meaning representation (concept) consisting of a set of words and phrases with similar semantic meaning, must be identified from that data. Some classes, such as those consisting of lists of names from a lexicon, are easy to specify, e.g., *city*, whereas others require a deeper understanding of language structure and the formal relationships (syntax) between words and phrases, e.g., *departure city*. A developer must supply this knowledge manually, or develop tools that automatically (or, semiautomatically) extract these concepts from annotated corpora with the help of language modelling tools.

Additionally, the lexical information is not specific to any domain. Rather, the entries attempt to capture what applies to the language at large, or represent specialized senses in a disjunctive manner. Note that semantic lexical knowledge is most sensitive to domain changes. Unlike syntactic constraints, semantic features tend to change as the word is used in different ways for different domains.

All these reasons raise the need for an automatic procedure. Clearly, an automatic or semi-automatic algorithm for extracting semantic classes from text can significantly reduce development-time in many natural language processing systems. Unsupervised induction of semantic classes is also the first step towards unsupervised learning of semantics from text, the "holy grail" of natural language

processing.

Our effort, for meaning identification, relied on the hypothesis that words that appear in similar lexical contents are semantically similar. The lexical environment was the only text's feature that was explored. In some sense, our approach tried to discover the meaning, which is hidden in a flexible use of natural language, without the use of strict syntactic rules. This is achieved by exploiting only the frequential aspect of data by using statistical and probabilistic techniques.

Outline

The rest of the thesis is organized as follows: Chapter 2 provides the necessary theoretical background and in Chapter 3 some related projects are presented. Chapter 4 presents in details our approach in the area of semantic class induction and in Chapter 5 the experimental procedure is being thoroughly described. In Chapter 6 an evaluation of the proposed system is provided and finally, in Chapter 7 some interesting conclusions are provided along with an outline of future work and ideas that may improve the performance of the proposed system.

Chapter 2

Theoretical Background

2.1 Importance of statistics

This chapter introduces some basic material on probabilities and statistics required for the understanding of this work. These definitions play central role in corpora processing since through their interpretation and implementation, epitomize the meaningful data. This ability can be compared with a dive in a deep and foggy lexical environment that results in discovery of the treasure of meaning. Corpora can be large collections of texts written by any editor or transcribed human - human, human - computer dialogues.

Obviously the desired meaning is hidden inside of each word. Also, each word can be viewed as a distinct event which whenever it occurs, it transmits, in a sense, some general information. The notion of "word" in linguistics is denoted by the term "lexeme". Thus a lexeme is the minimal unit of language, which has a semantic interpretation and embodies a distinct cultural concept. Furthermore, the extracted knowledge can be broader if we study a larger lexical field that consists of more than one lexeme. A single lexeme can be a "lexical unit", but in writing it is very common, more than one lexemes to behave as a lexical unit. So, is there any difference between lexeme and lexical unit or

not? Cruse[3] distinguishes lexemes from lexical units. The formers are the items listed in the lexicon, or ideal dictionary of a language. A lexeme corresponds to a particular word or word form, and can be associated with indefinitely many senses. The latters are form-meaning complexes with stable and discrete semantic properties, and the meaning component is called a sense, corresponding to the intuitive notion of sense. So *bank* is a lexeme, while *bank-financial institution* and *bank-edge of a river* are lexical units. We will see that for our work this approach does not propose clear criteria for establishing Cruse's distinction. We underline only the fact that it is myopic policy to isolate word from its lexical environment. It is reasonable to claim that a word preserves a kind of relationship with its neighboring words, in some way. Still the concept of "neighbor" remains abstract. We can guess correctly that it is worthless to treat a whole sentence as a single lexical unit that expresses a very specific meaning. Therefore, few consecutive words may form a lexical unit with a particular meaning. In such case we are sure that each word contains an amount of information about the other words of the lexical unit. This means that the probability of one word's appearance is not independent. In the opposite case the consecutive words would not form a lexical unit with a comprehensive meaning. These considerations trigger off the thought to find an alternative, single lexeme in order to express the same meaning that the whole lexical unit does. But still we need a metric to estimate the precision of this idea. Additionally, we cannot ignore the fact that a certain matter can be said or written using more than one lexical unit. Practically this is interpreted to the fact that people use many different lexical units to express approximately the same concept. So it is important to explore the semantic relationship between the lexical units of a corpus regardless of their place in the text.

2.1.1 Counting Words

Probability theory deals with predicting how likely it is that an event will take place. There are two interesting views of probabilities:

The objectivist view states that probabilities are real aspects of the world that can be measured by relative frequencies of outcomes of experiments. In contrast, according to the subjectivist view, probabilities are descriptions of an observer's degree of belief or uncertainty rather than having any external significance. These contrasting views are also referred to as Frequentist vs. Bayesian. Both views are relevant for linguistics; yet, the laws of probability theory remain the same under both interpretations.

Probabilities are based on counting things. In our case, statistical language processing requires computation of word probabilities. These probabilities are computed by counting words or lexical units in a corpus. The classical definition of probability, as given by Pascal is:

"*The probability of an event x is computed as the relative frequency with which x occurs in a sequence of n identical experiments.*"

So, the probability of a word w can be approximated by:

$$p(w) = \frac{occurences \quad of \quad word \quad w}{number \quad of \quad words} \tag{2.1.1}$$

which is the relative frequency with which w occurs in the corpus.

2.1.2 Punctuation Marks

Suppose that we have to count the words of the following sentence from Shakespeare's Hamlet and compute the probability of the word "God".

"Oh God, I could be bounded in a nutshell and count myself a king of infinite space."

If we count punctuation marks as words, the sentence has totally 19 words, if we do not 17 words..

Whether we count punctuation marks as words depends on the task. Tasks such as grammar checking or author-identification must treat punctuation marks as words because in these cases the location of the punctuation is important. Usually corpora of spoken language do not have punctuation marks.
In our work punctuation marks are not taken under consideration.

2.2 N-grams Probabilistic Model

This model proposes the assignment of probabilities to strings of words. Based on this method we can easily compute the probability of an entire sentence or predict the next word in a sequence.

The simplest probabilistic version of this model allows every word to have the same probability of following every other word. A more robust model let every word follow every other word, with the appearance of the following word to be depended on its normal frequency of occurrence. We still consider the individual relative frequency of each word. The following example, based on real data, can verify the precision of this simplistic approach.

Brown corpus[8] has $1,000,000$ words. The word "the" occurs 69,971 times in the corpus and the word "rabbit" occurs 11 times. Thus the probabilities are 0.07 and 0.00001 for the words "the" and "rabbit", respectively. Suppose that we have just read this part of a sentence: "Just then, the white...". Furthermore, suppose that we are curious about what the next word will be. If we use the simple model, we will conclude that the word "the" is the most possible word to follow "white". But this seems totally false because there is no meaning in the sentence "Just then, the white the". Doubtless the sentence "Just then, the white

rabbit" sounds more reasonable. This example shows that the computation of the probability of word sequences must use the conditional probability of a word given the previous word. Particularly that means that the probability of a word given the previous one is higher than its probability otherwise.

2.2.1 Conditional probability and independence

The notion of conditional probability can be considered as a kind of updated probability of an event given some knowledge. The probability of an event before gaining additional knowledge is referred to as the prior probability of the event. The new probability of the event estimated using the additional knowledge is called posterior probability of the event. The event of interest is formed by the occurrences of a word in the corpus. Using symbol Ω we denote the sample space, which is discrete, having finite number of elements. The sample space, which corresponds to a corpus, includes all the occurrences of each distinct word of the corpus. That is, the sample space Ω includes all the events of the corpus. For instance, assume a small corpus: "A tiny corpus tiny". For the previous corpus: Ω = { occurrence of the word "A", occurrence of the word "tiny", occurrence of the word "corpus", occurrence of the word "tiny" for second time }.

The event of the occurrence of "tiny" in the corpus is denoted as tiny and is: tiny = {occurrence of the word "tiny", occurrence of the word "tiny" for second time}.

We define the probability of the occurrence of the word "tiny" according to Eq. (2.1.1) as:

$$p(tiny) = \frac{|tiny|}{|\Omega|} \tag{2.2.1}$$

where $|tiny|$ is the number of elements in the set "tiny" and $|\Omega|$ is the number of elements in the probability space Ω. Thus $p(\Omega) = 1$. So, the probability of the

event $p(tiny)$ is: $p(tiny) = \frac{|tiny|}{|\Omega|} = \frac{2}{4}$.

For the general case:

$$p(event) = \frac{|event|}{|\Omega|} \tag{2.2.2}$$

where each event is a subset of Ω.

The conditional probability of a word w_2 assuming that word w_1 has occurred ($p(w_1) > 0$), denoted $p(w_2|w_1)$ equals:

$$p(w_2|w_1) = \frac{p(w_2 \cap w_1)}{p(w_1)} \tag{2.2.3}$$

which can easily be transformed into:

$$p(w_2|w_1)p(w_1) = p(w_2 \cap w_1) \tag{2.2.4}$$

Eq. (2.2.3) gives:

$$p(w_1|w_2) = \frac{p(w_1 \cap w_2)}{p(w_2)} \tag{2.2.5}$$

We can do the conditionalization either way because set intersection is symmetric, $w_2 \cap w_1 = w_1 \cap w_2$. So Eq. (2.2.3) becomes:

$$p(w_2|w_1) = \frac{p(w_2)p(w_1|w_2)}{p(w_1)} \tag{2.2.6}$$

Two words w_1, w_2 are considered to be conditionally independent if $p(w_2 \cap w_1) = p(w_2)p(w_1)$.

Conditional probability and independence can be the basis for computing the probability of a string of words. A string of words can be represented as $w_1, w_2, ..., w_{n-1}, w_n$ or $w_{1..n}$. Assuming the occurrence of each word in the corpus as an independent occurrence, we can write the probability of a string of words as follows:$p(w_1, w_2, ..., w_{n-1}, w_n)$ or $p(w_{1..n})$.

Using the chain rule of probability we represent $p(w_{1..n})$ as:

$$p(w_{1..n}) = p(w_1)p(w_1|w_2)p(w_3|w_{1..2})...p(w_n|w_{1..n-1}) = \prod_{k=1}^{n} p(w_k|w_{1..k-1}) \tag{2.2.7}$$

Since there is not any easy way for computing the probability of a word given all the previous words, an alternative solution for this task is to find a satisfactory approximation. The bigram model proposed for solving this difficulty, assumes that the probability of a word depends only on the previous word. In other words, $p(w_n|w_{1..n-1})$ is approximated by the conditional probability of the word that preceded $p(w_n|w_{n-1})$. This approximation is referred as a Markov assumption. Markov models are probabilistic models, which predict a future event without much prior knowledge. In the case of bigram (first order Markov model) models they need to know only the preceding word.

It is obvious that the trigram (second order Markov model) model looks two words into the past. Generalizing bigrams and trigrams, N-grams are resulted by which the probability of a word given all the previous words can be approximated by the probability given only the previous N words.

$$p(w_n|w_{1..n-1}) \simeq p(w_n|w_{(n-N+1)..(n-1)}) \tag{2.2.8}$$

For a bigram grammar, $p(w_{1..n})$ can be found by combining Eq. (2.2.8) and Eq. (2.2.7):

$$p(w_{1..n}) \simeq \prod_{k=1}^{n} p(w_k|w_{k-1}) \tag{2.2.9}$$

2.2.2 Smoothing

For any particular corpus, it is possible that some N-grams not to exist in this corpus. The consequence is that the N-gram model assigns zero probability to these N-grams. Also, using only relative frequencies to estimate N-grams probabilities might produce poor estimates when the counts are too small. This major problem raises the need to find a way of reevaluating zero probability and low probability N-grams and assigning them non-zero values. This procedure is called

smoothing.

Add-One Smoothing

This algorithm suggests to take the bigram counts and before normalizing them to probabilities, to add one to all the counts. This algorithm is very simple and in practice does not perform well. However it stands as an introduction to the concept of smoothing that is implemented much better by other algorithms.

Considering the unsmoothed maximum likelihood estimate of the unigram probability:

$$p(w_x) = \frac{c(w_x)}{\sum_i c(w_i)} = \frac{c(w_x)}{N} \tag{2.2.10}$$

where $c(w_x)$ is the frequency (counts) of word w_x in the corpus and N represents the total number of word tokens in the corpus.

The basic idea of smoothing relies on the c's adjustment. The adjusted count for add-one smoothing is defined by adding one to the count c and then multiplying by the factor N/(N+V), which is a normalization factor. Then the adjusted count is:

$$c_i^* = \frac{(c_i + 1)N}{(N + V)} \tag{2.2.11}$$

where V is the vocabulary size of the corpus.

Eq. (2.2.11) can be turned into probabilities p_i^* by dividing with the total number of word tokens:

$$p_i^* = \frac{(c_i + 1)}{(N + V)} \tag{2.2.12}$$

Applying Eq. (2.2.12) to Eq. (2.2.10), the add-one-smoothed probability for a bigram is defined as:

$$p^*(w_n|w_{n-1}) = \frac{c(w_{n-1}w_n) + 1}{c(w_{n-1}) + V} \tag{2.2.13}$$

An alternative view of smoothing

Actually a smoothing algorithm discounts some non-zero counts. This is a way to find the probability mass, which will be assigned to the zero counts. An alternative way to refer to lowered counts c^* is to define a discount ratio d_c:

$$d_c = \frac{c^*}{c} \tag{2.2.14}$$

The choice of value one (1) which is added to the each count c is arbitrary. This affects the probability mass that is moved near the zero value. A solution to this problem is the choice of smaller values regarding the situation.

2.2.3 Witten-Bell Discounting

This algorithm is referred as Method C, a method initially introduced by Alistair Moffat [19] and is considered to perform better than Add-One smoothing. In [30], Witten and Bell surveyed and compared several approaches to the zero-frequency problem that have been used in text compression systems. Witten and Bell described the zero-frequency problem in the case of adaptive word coding assuming a coding scheme in which the encoder reads the next word of text, searches for it in a list and transmits an index extracted from the list in place of the word. If the next word is not appeared in the list, a special code, called escape code, must be transmitted followed by the unknown word. This new word is added to the encoder and decoder's lists in case it appears again. According to this method each word is assigned an associated frequency. The computing of probability of the escape character, by estimating the likelihood of a novel word occurring, can solve the zero-frequency problem.

Similarly, a novel N-gram could then be assigned the probability of seeing it for the first time. The basic idea behind this conception is to "use the count

of things we have seen once to help estimate the count of things we have never seen".

We can compute the probability of seeing a novel N-gram by counting the number of times we saw N-grams for the first time in the corpus. The count of the first-time seen N-grams is simply the number of N-gram types we have already seen.

Hence we can estimate the total probability mass of all the zero N-grams by dividing the number of N -gram types we have seen with the sum of number of tokens and the number of N -gram types we have seen:

$$\sum_{i:c_i=0} p_i^* = \frac{T}{N+T} \tag{2.2.15}$$

where T is the N-gram types we have already seen and N is the number of tokens.

Probability given by Eq. (2.2.15) is the total probability of unseen N-grams. This "amount of probability" needs to be divided in order to assign a part of it to each zero N-gram. A simple compromise is to divide equally. Letter Z denotes the total number of N-grams with count zero. So the equal share of the probability mass is:

$$p_i^* = \frac{T}{Z(N+T)} \tag{2.2.16}$$

The probability of all the seen N-grams is given by the equation:

$$p_i^* = \frac{c_i}{N+T'}, \quad if \quad c_i > 0 \tag{2.2.17}$$

Extending the Witten-Bell discounting to bigrams, the type-counts are conditioned on some history. The probability of seeing for first time a bigram w_{n-1} w_n is equivalent to the probability of seeing a new bigram starting with the word w_{n-1}.

According to the Eq. (2.2.14) the probability of a bigram $w_x\ w_i$ we have not seen is:

$$\sum_{i:c(w_x w_i)=0} p^*(w_i|w_x) = \frac{T(w_x)}{N(w_x)+T(w_x)} \tag{2.2.18}$$

where $T(w_x)$ is the number of bigram types on the previous word w_x we have already seen and $N(w_x)$ is the number of bigram tokens on the previous word w_x.

Distributing the probability mass of the Eq. (2.2.18) among all the unseen bigrams, we get:

$$p^*(w_i|w_{i-1}) = \frac{T(w_{i-1})}{Z(w_{i-1})(N+T(w_{i-1}))}, \quad \text{if} \quad c(w_{i-1}w_i)=0 \tag{2.2.19}$$

where $Z(w_{i-1})$ is the total number of bigrams with w_{i-1} as the first word, that have zero counts.

For the non-zero bigrams, we parameterize T on the history:

$$p^*(w_i|w_x) = \frac{c(w_x w_i)}{c(w_x)+T(w_x)}, \quad \text{if} \quad c(w_{i-1}w_i)>0 \tag{2.2.20}$$

2.2.4 Backoff

So far the algorithms we have presented have all made use of the frequency of an N-gram and have tried to compute the best estimate of its probability. In general N-grams that never appeared or appeared only few times, were given the same estimate. A reasonable extension of the previous methods (smoothing) is to try to build better estimates by looking at the frequency of the (N-1)-grams found in the N-gram.

If (N-1)-grams, found in the N-gram, are appeared rarely, then a low estimate is given to the N-gram. Otherwise, N-grams with (N-1)-grams of moderate frequency are given a higher probability estimate. This issue grounded in a more

general discussion deals with combining multiple probability estimates making use of different models. That is, if there are no examples of a particular trigram, let's say $w_{n-2}w_{n-1}w_n$, the computation of $p(w_n|w_{n-1}w_{n-2})$ can be achieved through the use of the bigram probability $p(w_n|w_{n-1})$. In the same manner, if we have no examples of $w_{n-1}w_n$ in order to compute $p(w_n|w_{n-1})$, we can use the unigram probability $p(w_n)$.

In the backoff model, as described above, an N-gram model is built based on a (N-1)-gram model. We only look to a lower-order N-gram if we have no examples of a higher-order N-gram. So the backoff model for the trigram $w_{i-2}w_{i-1}w_i$ can be calculated from:

Case 1:

$$p(w_i|w_{i-2}w_{i-1}) = p(w_i|w_{i-2}w_{i-1}) \quad \text{if} \quad c(w_{i-2}w_{i-1}w_i) > 0 \tag{2.2.21}$$

Case 2:

$$p(w_i|w_{i-2}w_{i-1}) = a_1 p(w_i|w_{i-1}) \quad \text{if} \quad c(w_{i-2}w_{i-1}w_i) = 0 \quad \text{and} \quad c(w_{i-1}w_i) > 0 \tag{2.2.22}$$

Case 3:

$$p(w_i|w_{i-2}w_{i-1}) = a_2 p(w_i) \quad \text{otherwise} \tag{2.2.23}$$

Parameters a_1 and a_2 are weighting factors, which ensure that the result of the previous equation system is a true probability. For the general case the form of backoff is:

$$\hat{p}(w_n|w_{(n-N+1)..(n-1)}) =$$

$$\tilde{p}(w_n|w_{(n-N+1)..(n-1)}) + \theta(p(w_n|w_{(n-N+1)..(n-1)}))a\hat{p}(w_n|w_{(n-N+2)..(n-1)}) \tag{2.2.24}$$

The θ notation indicates a binary function that selects a lower-order model only if the higher-order model produces a zero probability. Specifically, if $x = 0$ then $\theta(x) = 1$, else $\theta(x) = 0$. Each $p(.)$ is a Maximum Likelihood Estimation.

2.2.5 A clever combination: Backoff and Discounting

As was previously shown, discounting methods are used to calculate the probability mass which is assigned to unseen events, assuming that they were equally probable. Combining discounting with backoff, this probability can be distributed more cleverly.

Consider the following example, which shows how backoff can lead to probability greater than 1: Using relative frequencies, $\sum_{i,j} p(w_n|w_i w_j) = 1$, which means that the probability of a word w_n over all N-gram contexts equals to 1. If we use backoff in this case, adopting a lower order model, the probability of w_n will be greater than 1. So, discounting must be applied to backoff model.

Thus, the correct form of Eq. (2.2.23) is:

$$\hat{p}(w_n|w_{(n-N+1)..(n-1)}) =$$

$$\tilde{p}(w_n|w_{(n-N+1)..(n-1)})+\theta(p(w_n|w_{(n-N+1)..(n-1)}))a(w_{(n-N+1)..(n-1)})\hat{p}(w_n|w_{(n-N+2)..(n-1)}) \tag{2.2.25}$$

where $\tilde{p}(.)$ stands for the discounted MLE probabilities:

$$\tilde{p}(w_n|w_{(n-N+1)..(n-1)}) = \frac{c^*(w_{(n-N+1)..n})}{c(w_{1..(n-N+1)})} \tag{2.2.26}$$

Function a represents the amount of probability mass, which must be distributed from an N-gram to an (N-1)-gram:

$$a(w_n|w_{(n-N+1)..(n-1)}) = \frac{1 - \sum_\beta \tilde{p}(w_n|w_{(n-N+1)..(n-1)})}{1 - \sum_\beta \tilde{p}(w_n|w_{(n-N+2)..(n-1)})} \tag{2.2.27}$$

where β denotes $w_n : c(w_{(n-N+1)..(n-1)}) > 0$.

2.3 Vector Space Model

Vector space model primarily applies in the fields of information retrieval, a domain which is of great interest due to its widespread adoption of word-based

indexing and retrieval methods. Most current information retrieval systems are based on an extreme interpretation of the principles of compositional semantics. In these systems, the meaning of documents resides solely in the words that are contained within them. The ordering and constituency of the words that make up the sentences that make up documents play no role in determining their meaning. Because they ignore syntactic information, these systems are often referred to as bag of words methods.

In the vector space model, documents (and queries) are represented as vectors of features representing the terms that occur within them [24]. More properly, they are represented as vectors of features consisting of the terms that occur with the collection of documents, with the value of each feature indicating the presence or absence of a given term in a given document. These vectors can be denoted as follows:

$\vec{d} = (t_1, t_2, t_3, ..., t_N)$

$\vec{q} = (t_1, t_2, t_3, ..., t_N)$

where d represents a document and q a query.

In this notation, the various t features represent the N terms that occur in the document collection. The values that these features take on, represent the importance of these terms in each document and within the whole document collection. These are often referred to as term weights and can be determined in many ways. In this way, the document and query vectors mentioned earlier can be generalized as follows:

$\vec{d_j} = (w_{1,j}, w_{2,j}, w_{3,j}, ..., w_{n,j})$

$\vec{q_k} = (w_{1,k}, w_{2,k}, w_{3,k}, ..., w_{n,k})$

This characterization of individual documents as vectors of term weights allows to view the document collection as a matrix of weights, where $w_{i,j}$ represents

the weight of term i in document j. The columns of this weight matrix, called term-by-document matrix, represent the documents in the collection and the rows represent the terms.

2.3.1 Term Weighing Schemes

A typical and very simple approach for determining term weights is by a simple binary weighting scheme where values one or zero are applied indicating the presence or the absence of a given term in a document. Of course, a problem with the use of binary values is that it fails to capture the fact that some terms are more important to the meaning of a document than others.

In practice, the method used to assign weights in the documents and query vectors has an enormous impact on the effectiveness of a retrieval system. Two factors have proven to be critical in deriving effective term weights:

The first factor is the term frequency which is based on the simple notion that terms that occur frequently within a document may reflect its meaning more strongly than terms that occur less frequently and should thus have higher weights. The frequency of the term i in the document j is represented as: $tf_{i,j}$.

The second factor to consider is the distribution of terms across the collection as a whole. Terms that are limited to a few documents are useful for discriminating those documents from the rest of the collection. On the other hand, terms that occur frequently across the entire collection are less useful in discriminating among documents. The fraction $\frac{N}{n_i}$, where N is the total number of documents in the collection and n_i is the number of documents in which term i occurs, provides exactly this measure that favors terms that occur in fewer documents. Due to the large number of documents in many collections, this measure is usually squashed with a log function, which leads to the final inverse document frequency of the

term i in the document collection: $idf_i = log(\frac{N}{n_i})$.

Combining the term frequency with the inverse document frequency results in a weighting scheme known as $tf \times idf$ weighting scheme: $w_{i,j} = tf_{i,j} \times idf_{i,j}$; that is the weight of term i in the vector for document j is the product of its overall frequency in j with the log of its inverse document frequency in the collection.

2.4 Contextual Word Similarity

The concept of word similarity was traditionally captured with thesauri. A thesaurus is a lexicographic resource that specifies semantic relationships between words, listing for each word related words such as synonyms, hyponyms and hypernyms. Thesauri have been used to assist writers in selecting appropriate words and terms and in enriching the vocabulary of a text. To this end, modern word processors provide a thesaurus as a built in tool.

The area of information retrieval has provided a new application for word similarity in the framework of query expansion. Good free-text retrieval queries are difficult to formulate since the same concept may be denoted in the text by different words and terms. Query expansion is a technique in which a query is expanded with terms that are related to the original terms that were given by the user, in order to improve the quality of the query. Various query expansion methods have been implemented, both by researchers and in commercial systems that rely on manually crafted thesauri or on statistical measures for word similarity.

Word similarity may also be useful for disambiguation and language modeling in the area of NLP and speech processing. Many disambiguation methods and language models rely on word co-occurrence statistics that are used to estimate the likelihood of alternative interpretations of a natural language utterance (in

speech or text). Due to data sparseness, though, the likelihood of many word co-occurrences cannot be estimated reliably from a corpus, in which case statistics about similar words may be helpful.

It should be noted that while all the applications mentioned above are based on some notion of "word similarity" the appropriate type of similarity relationship might vary. A thesaurus intended for writing assistance should identify words that resemble each other in their meaning, like "aircraft" and "airplane", which may be substituted for each other. For query expansion, on the other hand, it is also useful to identify contextually related words, like "aircraft" and "airline", which may both, appear in relevant target documents. Finally, co-occurrence-based disambiguation methods would benefit from identifying words that have similar co-occurrence patterns. These might be words that resemble each other in their meaning, but may also have opposite meanings, like "increase" and "decrease".

2.4.1 Distance Functions

In order to calculate the resemblance between words, a function is needed to measure the similarity between the words' distributions. This function, usually referred to as distance function when it computes the dissimilarity between words or similarity function when it computes the similarity is defined as follows:

A metric space is a set X together with a function d (called a "metric" or "distance function") which assigns a real number $d(x, y)$ to every pair $x, y \epsilon X$ satisfying the properties:

1. $d(x, y) \geq 0$ and $d(x, y) = 0 \Leftrightarrow x = y$,
2. $d(x, y) = d(y, x)$,
3. $d(x, y) + d(y, z) \geq d(x, z)$

The last property is known as the "triangle inequality".

2.4.2 Vector Product Similarity Measure (Cosine Measure)

This geometric metric is a similarity measure, rather than a difference measure. It is related to the angle between two vectors; the "closer" two vectors are, the smaller the angle between them.

Let Y be a random variable taking values in Υ. Suppose we are considering exactly two hypotheses about Y: Y is distributed according to $q(H_q)$, and Y is distributed according to $r(H_r)$. The Vector Product, VP, similarity between the two distributions is calculated as follows:

$$VP(q,r) = \frac{\sum_{y\epsilon\Upsilon} q(y)r(y)}{\sqrt{\sum_{y\epsilon\Upsilon} q(y)^2 \sum_{y\epsilon\Upsilon} r(y)^2}} \tag{2.4.1}$$

Notice that the cosine measure is an inverse distance function, in that it achieves an upper bound of 1 when $q(y) = r(y)$ for all y.

2.4.3 Measuring Similarity in Bigram Probabilistic Model

In order to calculate the similarity between two lexical units according to their lexical environment, we have to take into account left and right contexts. Let's consider a word w with its neighbors in a word sequence:

$$\ldots \quad v_{1,L} \quad w \quad v_{1,R} \quad \ldots$$

with $v_{1,L}$ representing the word in the left context and $v_{1,R}$ representing the word in the right context. Two probability distributions are calculated, $p(v_{1,L}|w)$ and $p(v_{1,R}|w)$, for the left and right contexts respectively. The right-context bigrams are calculated using the usual word order, and the left-context bigrams are calculated with a reversed-order corpus.

In order to estimate the similarity of two words, w_1 and w_2, the sum of the symmetric left and right context-depended distances is needed. So, the total

distance, D, between the probability distributions for w_1 and w_2 is :

$$D^{LR}(w_1, w_2) = D^L_{12} + D^L_{21} + D^R_{12} + D^R_{21} \tag{2.4.2}$$

where D^L and D^R are the left-context and right-context distance respectively and D_{12} denotes the (possibly symmetric) distance between w_1 and w_2 [22, 25].

In order to calculate the semantic distance between two words, the cosine distance between two feature vectors is computed; each feature vector of a word w measures the conditional probability of all possible contexts v_i given that word $p(v_i|w)$, i.e., each vector contains bigram language model probabilities for (context, word) pairs. Given the symmetric nature of the Vector Product metric, $VP^L_{12} \equiv VP^L_{21}$ and $VP^R_{12} \equiv VP^R_{21}$, Eq. (2.4.2) becomes:

$$D^{LR}(w_1, w_2) = VP^{LR}(w_1, w_2) = VP^L_{12} + VP^R_{12} \tag{2.4.3}$$

where:

$$VP^L_{12} = \frac{\sum_{v_{1,L}\epsilon V} p^L_1(v_{1,L}|w_1)p^L_2(v_{1,L}|w_2)}{\sqrt{\sum_{v_{1,L}\epsilon V} p^L_1(v_{1,L}|w_1)^2 \sum_{v_{1,L}\epsilon V} p^L_2(v_{1,L}|w_2)^2}} \tag{2.4.4}$$

and

$$VP^R_{12} = \frac{\sum_{v_{1,R}\epsilon V} p^R_1(v_{1,R}|w_1)p^R_2(v_{1,R}|w_2)}{\sqrt{\sum_{v_{1,R}\epsilon V} p^R_1(v_{1,R}|w_1)^2 \sum_{v_{1,R}\epsilon V} p^R_2(v_{1,R}|w_2)^2}} \tag{2.4.5}$$

where $V = (v_1; v_2; ...v_N)$ is the vocabulary set, and $p^L_1(v_{i,L}|w_1)$ is the conditional probability of word v_i preceding w_1 in the corpus given word w_1, i.e., the v_i, w_1 bigram model probability.

2.4.4 Measuring Similarity in Vector Space Model

Having determined the term weights, a ranking function is needed to measure the similarity between documents. Vector Space model conceives of the features used to represent documents as dimensions in a multi-dimensional space. Correspondingly, the weights that serve as values for those features serve to locate

documents in that space. Documents that are located close to each other can then be said to be more similar than documents that are farther away.

In general, in order to compute the similarity between two vectors, dot product between vectors is used as shown below.

$$s(\vec{d_k}, \vec{d_j}) = \vec{d_k} \cdot \vec{d_j} = \sum_{i=1}^{N} w_{i,k} \times w_{i,j} \tag{2.4.6}$$

The dot product between two vectors is not particularly useful as a similarity metric, since it is too sensitive to the absolute magnitudes of the various dimensions. This issue can be dealt with by normalizing the document vectors converting all of them to a standard length. In this way, the importance of the exact length of a document's vector in the space is eliminated, emphasizing instead on the direction of each document vector.

This leads to the similarity measure, known as cosine measure, which determines the angle between the document vectors. The similarity between document D_i and D_k is defined as:

$$s(\vec{D_i}, \vec{D_k}) = \frac{\sum_{j=1}^{V} w_{D_i,j} \times w_{D_k,j}}{\sqrt{\sum_{j=1}^{V} w_{D_i,j}^2 \times \sum_{j=1}^{V} w_{D_i,j}^2}} \tag{2.4.7}$$

where $w_{D_i,j}$ is the weight of term j in document D_i and V represent the vocabulary. The denominator in this equation, called the normalization factor, discards the effect of document lengths on document scores.

2.5 Combining Metrics

Information retrieval (IR) systems are based either directly or indirectly, on models of the retrieval process. These retrieval models specify how representations of text documents and information needs should be compared in order to estimate the likelihood that a document will be judged relevant.

As these retrieval models were being developed, many experiments were carried out to test the effectiveness of these approaches. In these experiments, it was observed that different retrieval models, or alternatively, variations on ranking algorithms, had surprisingly low overlap in the relevant documents that were found, even when the overall effectiveness of the algorithms was similar [18]. It was also shown that the practice of searching on multiple document representations was more effective than searching on a single representation [10] suggesting that finding all the relevant documents for a given query was beyond the capability of a single simple retrieval model or representation.

The lack of overlap between the relevant documents found by different ranking algorithms and document representations led to two distinct approaches to the development of IR systems and retrieval models. One approach has been to create retrieval models that can explicitly describe and combine multiple sources of evidence about relevance; the other approach has been to design systems that can effectively combine the results of multiple searches based on different retrieval models.

The motivation for both these approaches is to improve retrieval effectiveness by combining evidence. Combining multiple, heterogeneous searches is the basis of the "meta search" engines on the Web (e.g., MetaCrawler). In general, selecting the "best" ranking algorithm is not necessarily the ideal choice, since potentially valuable information may be wasted by discarding the results of less successful classifiers. This observation motivates the concept of "combining" wherein the outputs of all the available classifiers are pooled before a decision is made. This approach is particularly, useful for difficult problems such as those the involve a large amount of noise, limited number of training data, or unusually high dimensional patterns.

Among many strategies, the most commonly used combing strategy is by simply averaging the output of the classifiers. It has been observed [29] that the simple combining strategies are best suited for situations where the classifiers all perform the same task (which is the case for IR), and have comparable success. Simple combination strategies can fail when even one of the classifiers being combined has very poor performance or very uneven performance. In such cases, using different weights for each classifier improves the effectiveness of the combined metric. Classifiers that provide less evidence about relevance are given lower weights in the combination to improve overall performance.

In our approach, we have two different forms of the vector product metric with the same objective but with different perspective of the input data: one metric with broader lexical scope and another one that concentrates on the immediate context of each word. Our goal was to create a combined metric that takes into account both wide-context and narrow context information.

2.6 Synopsis

In this chapter some basic material on probability and statistics was introduced. The notion of "lexical unit" and the reason for which probability theory pays a central role in corpora processing were explained. Next we saw the use of N-grams in natural language modelling followed by the Vector Space model. The concept of word similarity was also presented along with distance functions and the way they apply to bigram and vector space model. Lastly, an introduction was made regarding the combination of metrics and the way that it applies in our approach.

Chapter 3

Related Work

3.1 Introduction

Numerous techniques and systems have been proposed in the area of induction semantic classes from textual data. Among them, three major families of techniques can be distinguished each one with different perspective but with the same objective. These families are: numerical, symbolic and hybrid.

Numerical approaches exploit the frequential aspect of data, and use statistical and probabilistic techniques. Symbolic approaches examine the structure of data taking mainly under consideration the syntax of the textual data. Finally, hybrid approaches are combinations of numerical and symbolic techniques.

3.2 Overview of Related Projects

Among the numerical approaches, [25] uses a semi-automatic approach in order to cluster words according to a similarity metric, working in a domain-specific corpus, ATIS. However, the resulting classes had to be hand-revised. More recently, in [21, 22], an automatic procedure is described that classifies words and

concepts into semantic classes, according to the similarity of their lexical environment. This approach induces semantically compact classes especially for restricted domains where the expressive style is oriented towards the specific needs of the certain task. Among symbolic approaches, Text-to-Onto [16, 14, 15], deals with discovering non taxonomic relationships from text and enhancing an already defined taxonomic hierarchy. The Text-to-Onto system uses shallow parsing as a natural language module. Asium [7, 6], is able to learn semantic knowledge from text by (mostly) extracting concepts/classes and putting them into taxonomic relationships. It is a semi-automatic system, meaning that user's control is needed in the process. In [13], a system is described that constructs a domain specific ontology from text documents [27]. The documents are read, processed, and a graph-structured ontology is produced using contemporary statistical methods of information retrieval such as Boolean, extended Boolean and Vector Space approaches.

In the next paragraphs, the projects that were just mentioned are presented thoroughly.

3.2.1 "Semi-automatic acquisition of domain-specific semantic structures."[25]

This work deals with the semi-automatic induction of a grammar from unannotated corpora belonging to a restricted domain. The resultant grammar contains language structures that may be semantic, syntactic or a tight coupling of both, which are all conducive towards language understanding. An iterative clustering algorithm is proposed, having the inferred grammar hand-revised at every iteration for quality improvement. This constitutes the semi-automatic nature of this approach. In their experiments, they used the training and test sets of

the domain specific ATIS (Air Travel Information System) corpus which contains transcribed utterances dealing with travel information.

In their work, a statistical approach is adopted in order to accomplish the understanding of natural language. The iterative procedure proposed to cluster the words from a corpus is implemented both spatially and temporally. In spatial clustering, words or multi-words entities with similar left and right linguistic contexts are clustered together. Consider for example the clustering of the entities e_1 and e_2. If p_1 and p_2 denote the unigram distributions of words occurring to the left of e_1 and e_2 respectively, then the similarity of the two distributions can be measured by the divergence metric Div (or symmetrized Kullback-Leiber distance) as shown below:

$$Div(p_1^{left}, p_2^{left}) = D(p_1^{left} \parallel p_2^{left}) + D(p_2^{left} \parallel p_1^{left}) \tag{3.2.1}$$

where:

$$D(p_1 \parallel p_2) = \sum_{i=1}^{V} p_1(i) log \frac{p_1(i)}{p_2(i)} \tag{3.2.2}$$

and V denotes the corpus' vocabulary.

When both the left and right contexts are considered, then the distance metric Dist derives from the following equation:

$$Dist(e_1, e_2) = Div(p_1^{left}, p_2^{left}) + Div(p_2^{right}, p_1^{right}) \tag{3.2.3}$$

The probabilities are obtained from the frequency counts in the corpus and in order to avoid sparse data problems only words that have at least M occurrences are taken under consideration. The proposed algorithm selects the N most similar pairs (i.e. lowest values for Dist) to form spatial clusters that are labelled as SC_i, where i is a counter of the number of spatial clusters formed. Then, the appropriate word pairs in the corpus are substituted by their SC labels.

Spatial clustering is followed by temporal clustering where words or multi-word entities that co-occur sequentially are clustered together. At this point, Mutual Information (MI) is used as the metric for clustering as show in the following equation:

$$MI(e_1, e_2) = P(e_1, e_2) log \frac{P(e_2|e_1)}{P(e_2)} \tag{3.2.4}$$

Again, only words that have at least M occurrences are taken under consideration and the N pairs of entities with highest MI are selected to form temporal clusters labeled as TC_j, where j is a counter of the number of temporal clusters formed. Thereafter, the appropriate word pairs are substituted by their TC labels and the algorithm proceeds to another iteration of spatial clustering.

In this way, this agglomerative clustering approach produces a context-free grammar, where the SCs and TCs are the non-terminal of this grammar. SC clusters tend to be semantic structures and TC clusters tend to be phrasal structures. The grammar then is post-processed by hand-editing and the procedure starts from the beginning.

3.2.2 "Auto-Induced Semantic Classes"[21, 22]

In this work, an automatic procedure is described that classifies words and concepts into semantic classes according to the similarity of their lexical environment. This approach induces semantically compact classes especially for restricted domains where the expressive style is oriented towards the specific needs of the certain task.

They use an unsupervised training approach consisting of two complementary procedures. First, they use n-gram statistics to determine the similarity of words and more generally, phrases, by looking at their bigram lexical contexts within

a single domain. According to their research, they compared four different similarity metrics in order to accomplish auto-inducement of semantic classes. These metrics are the Kullback-Leibler distance, the Information-Radius distance, the Manhattan-Norm distance, and the Vector-Product [1, 4, 5, 17, 22, 23]. Phrases that are determined to be the most similar are grouped into the same semantic class (or, concept). The performance of the aforementioned metrics was evaluated using four different application domains: a movie information retrieval service, the Carmen-Sandiego computer game, a travel reservation system, and the Wall Street Journal (WSJ) corpus. The first three domains used relatively small, transcribed dialogues between human subjects and agents while the WSJ was a large, text-based corpus. Of the first three domains, the computer game was a Wizard-of-Oz scenario, whereas the movie and travel information services were human-human dialogues.

In their approach, words or phrases are paired according to the similarity of their syntactic environments. A candidate word, w, is considered with its nearest neighbors in a word sequence:

$$\{\ldots \quad u_1^L \quad w \quad u_1^R \quad \ldots\} \tag{3.2.5}$$

with u_1^L representing the first words in the left context and u_1^R representing the first word in the right context. Two probability distributions are calculated, $p^L(u_1^L|w)$ and $p^R(u_1^R|w)$ for the left and right contexts respectively. The right-context bigrams are calculated using the usual word order, and the left-context probabilities are calculated with a reversed-order training corpus using standard n-gram training tools. They used the 1996 version of the CMU toolkit to calculate the n-gram statistics [2]. The similarity of two words, w_1 and w_2 is estimated as the sum of the symmetric left and right context-dependent distances [25]. Then,

the total distance between the probability distributions for these two words is calculated as:

$$d^{LR}(w_1, w_2) = D^L_{12} + D^L_{21} + D^R_{12} + D^L_{21} \tag{3.2.6}$$

where $D^L_{12}(w_1, w_2) \equiv D(p^L_1 \parallel p^L_2)$ is the left-context distance for a given metric D. The '$\parallel$' symbol refers to the distance, as calculated using some metric, between the two given conditional probability distributions. The conditional probability terms are of the form: $p^L_1 \equiv p^L_1(u^L_1|w_1)$. This is the probability that the word u^L_1 precedes (is to the left of) the word w_1. The D^R distance terms are similar, using the right-context probabilities p^R as in the term, $p^R_2 \equiv p^R_2(u^R_1|w_2)$ which is the conditional probability that the word u^R_1 follows (is immediately to the right of) the word w_2.

Kullback-Leibler Distance (KL)

The total symmetric KL distance, for bigram lexical contexts, is given by Eq. (3.2.6). For two candidate words w_1 and w_2 the right, bigram-context distance KL^R_{12} is defined over the vocabulary V as:

$$KL^R_{12} \equiv KL(p^R_1(w_1)|p^R_2(w_2)) = \sum_{u^R_1 \epsilon V} p^R_1(u^R_1|w_1) log \frac{p^R_1(u^R_1|w_1)}{p^R_2(u^R_1|w_2)} \tag{3.2.7}$$

where the sum is over all the words in the vocabulary, V.

Information Radius (Shannon-Jannsen distance) (IR)

The IR distance is similar to the KL distance. The total symmetric IR distance $IR^{LR}(w_1, w_2)$ is given by Eq. (3.2.6). For two candidate words w_1 and w_2 the left, bigram-context distance IR^L_{12} is defined over the vocabulary V as:

$$IR^L_{12} = \sum_{u \epsilon V} p^L_1(u|w_1) log \frac{p^L_1(u|w_1)}{\frac{1}{2}(p^L_1(u|w_1) + p^L_2(u|w_2))} \tag{3.2.8}$$

Manhattan-norm distance (M)

The Manhattan-norm $M^{LR}(w_1, w_2)$ for two candidate words w_1 and w_2 is calculated as:

$$M^{LR}(w_1, w_2) = M_{12}^L + M_{12}^R \tag{3.2.9}$$

The left-context dependent term is:

$$M_{12}^L = \sum_{u \epsilon V} |p_1^L(u|w_1) - p_2^L(u|w_2)| \tag{3.2.10}$$

where $M_{12} \equiv M_{21}$.

Vector product similarity (VP)

This metric is a similarity measure rather than a difference measure. The total distance $VP^{LR}(w_1, w_2)$ for two candidate words w_1 and w_2 is calculated as:

$$VP^{LR}(w_1, w_2) = VP_{12}^L + VP_{12}^R \tag{3.2.11}$$

where the left-context vector product is:

$$VP_{12}^L = \frac{\sum_{u \epsilon V} p_1^L(u|w_1) p_2^L(u|w_2)}{\sqrt{\sum_{u \epsilon V} p_1^L(u|w_1)^2 \sum_{u \epsilon V} p_2^L(u|w_2)^2}} \tag{3.2.12}$$

They propose an iterative procedure for automatically inducing semantic classes, consisting of three main components: a lexical phraser, a semantic generalizer, and a corpus reparser as shown in figure 3.1.

First, the lexical phraser groups words in a single lexical unit. Next, a semantic generalizer generates rules that map words (and concepts) to concepts. Finally, a corpus parser re-parses the corpus using the rules generated from the semantic generalizer.

Lexical Phraser

This module generates a list of the most commonly co-occurring lexical phrases or sentence fragments. The lexical phraser groups consecutive words into phrases

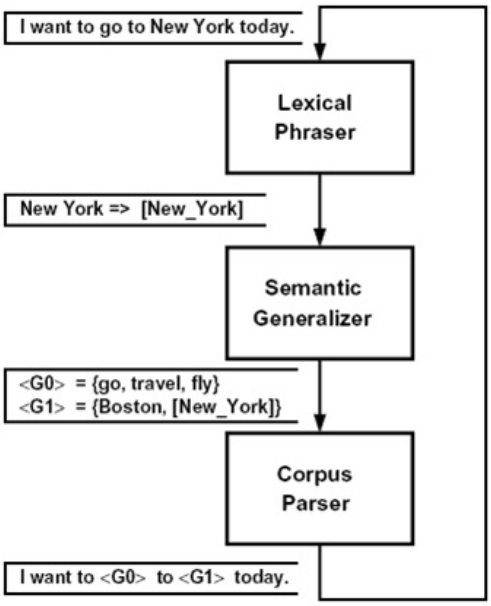

Figure 3.1: Iterative procedure used for the auto-induction of semantic classes. The sample sentence is from the Travel domain

by using a weighted point-wise mutual information (MI) measure to find those lexical entities that co-occur often. The n phrases with the highest MI measure,

$$MI(w_1, w_2) = p(w_1, w_2) log \frac{p(w_1, w_2)}{p(w_1)p(w_2)} \tag{3.2.13}$$

for the words w_1 and w_2 are kept at each iterations. They are only retained in successive iterations if they are classified into semantic groups in the following, semantic generalizer, module.

Semantic Generalizer

During this procedure, grammar rules are generated, where a rule maps a word or a previously formed class into a semantic class whose members share the same meaning. The main criterion for generating such groupings is the lexical or semantic similarity of the left of right-hand context for the members of a group.

Corpus Parser

At this point, the corpus is reparsed and all instances of each of the generalized phrases are replaced with the appropriate class. This is done for each of the grammar rules that were generated in the previous module. Then, the system stars again from the beginning.

3.2.3 Text-to-Onto[16, 14, 15]

In their work they describe an approach to discover non-taxonomic conceptual relations from text, built on shallow text processing techniques. They propose a generalized association rule algorithm that does not only detect relations between concepts, but also determines the appropriate level of abstraction at which to define relations. Their approach has been implemented on top of a shallow text processor for German adapted to the tourism domain called SMES (Saarbrucken Message Extraction System).

The architecture of SMES comprises a tokenizer based on regular expressions, a lexical analysis component including a word and a domain lexicon and a chunk parser.

Tokenizer

Its main task is to scan the text in order to identify boundaries of words and complex expressions like "$20.000" and to expand abbreviations.

Lexicon

The lexicon contains more than 120,000 stem entries and more than 12,000 subcategorization frames describing information used for lexical analysis and chunk parsing. Furthermore, the domain specific part of the lexicon associates word stems with concepts.

Lexical Analysis

It uses the lexicon to perform:

- *Morphological Analysis*

  In German language, compounds are extremely frequent and, hence, their analysis into their parts, e.g. "database" becoming "data" and "base", is crucial and may yield interesting relationships between concepts. Furthermore, morphological analysis returns possible readings for the words concerned, e.g. the noun and the verb reading for a word like "man" in "The old man the boats."

- *Recognition of name entities*

  Processing of named entities includes the recognition of proper and company names like "Hotel Schwarzer Adler" as single, complex entities, as well as the recognition and transformation of complex time and date expressions into a canonical format, e.g. "January 1st, 2000" becomes "1/1/2000".

- *Retrieval of domain-specific information*

  This step associates single words or complex expressions with a concept if a corresponding entry in the domain-specific part of the lexicon exists. E.g., the expression "Hotel Schwarzer Adler" is associated with the concept Hotel.

- *Part-of-speech tagging*

  Part-of-speech tagging disambiguates the reading returned from morphological analysis of words or complex expressions using the local context.

Chunk Parser

SMES uses weighted finite state transducers to efficiently process phrasal and sentential patterns. The parser works on the phrasal level, before it analyzes the overall sentence. Grammatical functions (such as subject, direct-object) are

determined for each dependency-based sentential structure on the basis of subcategorizations frames in the lexicon.

Dependency Relations

The primary output derived from SMES consists of dependency relations found through lexical analysis(compound processing) and through parsing at the phrase and sentential level. It is important for their approach that on these levels syntactic dependency relations coincide rather closely with semantic relations that are often found to hold between the very same entities.

Heuristics

Chunk parsing such as performed by SMES still returns many phrasal entities that are not related within or across sentence boundaries. This however means that the system misses many relations that often occur in the corpus, but are not detected due to the limited capabilities of SMES. Therefore, the SMES output has been extended to include heuristic correlations beside linguistics-based dependency relations.

Generally, their learning algorithm is based on the algorithm for discovering generalized association rules proposed by Srikant and Agrawal[26]. This algorithm finds associations that occur between items, e.g. supermarket products, in a set of transactions, e.g. customers' purchases, and describes them at the appropriate level of abstraction, e.g. "snacks are purchased together with drinks" rather than "chips are purchased with beer" and "peanuts are purchased with soda".

3.2.4 ASIUM[7, 6]

Asium is able to learn semantic knowledge from text. In this context it means extracting concepts/classes and putting them into taxonomic relationship. It is a

semi-automatic system meaning that user's control is needed in the process. As preliminary experimentation, Asium was applied to a very narrow and specific domain that contains cooking recipes. Asium learns semantic knowledge and ontologies in the form of subcategorization frames of verbs. A sub-categorization frame in this context is defined as: *<verb> <preposition or syntactic role: headword> <preposition or syntactic role: headword>*.

The ontology extracted represents generality relations between concepts in the form of an acyclic oriented graph. The concepts of this ontology are related only in an IS-A form.

For example, a sub-categorization frame of the sentence: *My father travels by car* is: *<to travel> <subject: father> <by: car>*. The system uses a stop list and it only takes headwords into consideration. So all articles, adjectives, etc. (*a, the, my, your, nice, beautiful, etc.*) are considered noise and are ignored, due to them still being believed to be preserved semantic information. Moreover, syntactic parser Sylex identifies whether headwords are expressions, i.e. *double decker, Ford Escort* or single words. The syntactic parser gives all possible frame interpretations of sentences and ASIUM uses all of them for this approach to avoid a very time consuming hand disambiguation step while still giving a good outcome.

Once each sentence has been instantiated into a frame the learning component takes them as input and learns an ontology. This step incorporates unsupervised clustering (bottom-up) and relies on the following assumption:

Headwords occurring after the same preposition or syntactic role, and with the same verbs represent the same concept.

For example from *<to travel> <subject: father> <by: car>* and *<to travel> <subject: father> <by: train>* one can conclude that *car* and *train* represent the

same concept, i.e. *motorized vehicle.*

This assumption is implemented in two steps. The first step gathers headwords that occur in the same contexts such as with the same verb and the same preposition or syntactic role. The second one builds synthetic frames according to verbs of subcategorization frames and assigns number of occurrence in the given context. For example, from the following instantiated frames:

<to travel> <subject: father> <by: car> <to travel> <subject: mother> <by: train>

<to drive> <subject: friend> <object: car>

<to drive> <subject: colleague> <object: motorbike>

<to drive> <subject: friend> <object: motorbike>

Asium might create synthetic frames, one per verb:

<to travel> <subject: [father(1), mother(1)]> <by: [car (1), train(1)]>

<to drive> <subject: [friend(2), colleague(1)]> <object: [car(1), motorbike(2)]>

At this stage clustering comes into play. The clustering is based on the distance between two clusters. In this context a cluster is represented as a list of headwords, for example *[car(1), train(1)].* Overlapping clusters are aggregated into a new cluster. Thus, clusters that contain the same headwords with the same frequencies are considered to be similar - their distance is zero. On the other hand, the distance of clusters that do not share any headword is the highest, equal to 1. The clustering algorithm is very simple and could be briefly described as an examination of each possible couple of clusters, aggregating those pairs that are the most similar(a threshold value is used for distance pruning) and repeating the same step over and over again until it is not able to aggregate any pair anymore. It is important to understand that aggregated are the headwords

of two clusters (no frames). For example *[car(1), train(1)]* and *[car(1), motorbike(2)]* might be aggregated to form new cluster called *motorized vehicles.* After the aggregation, the new cluster is propagated through all the synthetic frames, meaning that every occurrence of *[car(1), train(1)]* and *[car(1), motorbike(2)]* will be replaced with *motorized vehicle.* At this stage, a user is asked to accept or reject the aggregation to be propagated. The participation of the user is needed in such a method, not only to control the generality level of restrictions in verb frames but also to interactively correct the clusters in case of noise. For instance aggregation might yield new cluster *[car(1), train(1), bike(1), motorbike(2)].* While this cluster is certainly good for a frame <*to travel*>, it is no good for <*to drive*> since everyone knows that a bike is not drivable because it is not a motorized vehicle.

In their approach a concept/class of a building ontology is a cluster. Therefore at each level of clustering new classes are introduced. One can observe that clustering, which at each level creates only pairs, might lead to enormous number of useless classes. Asium has however a post-processing phase in which it removes all useless classes. This approach might be a big help in ontology construction in a narrow specific domain but it might not be very useful in a general one.

3.2.5 "Ontology Extraction from text documents by Singular Value Decomposition."[13]

This project is concentrated on statistical analysis methods, as compared to heuristic and rule based methods because of their simplicity and because these methods are based on fairly precise mathematical foundation. This system constructs a domain specific ontology from text documents, presented to it as a training set with a small amount of user feedback. The documents are read,

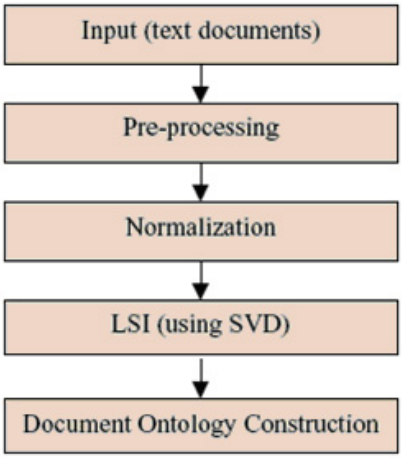

Figure 3.2: Proposed Architecture

processed, and a graph-structured ontology is produced.

In the process of achieving this goal, contemporary statistical methods of information retrieval such as Boolean, extended Boolean and vector space approaches have been used. Of all these approaches, vector space approach has been found to be the most efficient method. It describes each document as a set of terms. This set defines the document space such that each distinct term represents one dimension in that space. Each document in this document space is defined by the weights, or in other words frequencies, of the terms that represent it.

The architecture of the proposed system is briefly presented in fig. 3.2:

The input for this system is a set of text documents. These documents are pre-processed, a step that involves various procedures to facilitate meaningful statistical analysis and only meaningful terms are extracted for which their frequency is accurately counted. Then, for each word obtained at the previous step a normalized weight is calculated which leads to the creation of a term-document

matrix that describes the occurrence of meaningful terms in each document of the collection. This matrix is referred to as term-document matrix and contains terms as rows and documents as columns.

At the next stage, a set of concepts are determined from the previously extracted term-document matrix, where a concept is defined as a set of related terms. This is accomplished by using a method called Latent Semantic Indexing (LSI) which primarily involves decomposing the term-document matrix using Singular Value Decomposition (SVD). LSI is a statistical method that links terms into a useful semantic structure without syntactic or semantic natural language analysis and without manual human intervention. By using this method, each document is represented not by terms but by concepts that are truly statistically independent in a way that terms are not.

Singular value decomposition, is a method for matrix decomposition. It decomposes a matrix A, i.e. the m x n term-document matrix, with m terms and n documents, as $A = U * S * V^T$ where U is m x r matrix called the term matrix, V is r x n matrix called the document matrix and S is r x r diagonal matrix containing singular values of A in its diagonal in descending order. In LSI a new matrix is formed $A^q = U^q * S^q * (V^q)^T$ where A^q is derived from A by removing all but the largest q singular values, U^q is derived from U by removing all but the q columns corresponding to the remaining singular values and V^q is derived from V by removing all but q corresponding rows where $q <=$r. The matrix U^q is an m x n matrix representing correlations between terms in the document collection. Each column of this matrix is a vector which is considered to represent a concept.

The construction of the document ontology is accomplished by building concept nodes and term nodes from the term matrix (U) and document matrix (V) which have been obtained from SVD. A concept node represents a concept and

contains information about its concept name, terms that belong to that concept and their weights in that concept. The name of a concept is generated automatically and is a hyphenated string of the five most frequent terms in that concept. Since this does not produce excessively intuitive concept names, the user has the ability to modify the concept name, to a more meaningful one. A term node represents a term and contains information about its term name, concept that belongs to, and its weight in different concepts. The ontology constructed is a bipartite graph containing only two types of nodes: concepts and terms. Concept nodes are connected to term nodes but are not connected to other concept nodes directly. Term nodes are connected to other term nodes, only by being connected to a common concept node.

3.3 Synopsis

In this chapter, some related to our work projects have been presented. It was made clear that the techniques and systems that have been proposed in the area of semantic class induction from textual data are numerous. Each system, concentrates on specific view of the textual data but has as final goal the extraction of structure from text.

Chapter 4

Our Approach

4.1 Overview

Our approach lies on an iterative procedure for automatic induction of semantic classes, consisting of two main components: a *class generator* and a *corpus parser*. The *class generator*, explores the context information of every word, calculating the similarity between words and groups semantically similar words or concepts into classes. The semantic similarity distance combines two variations of the Vector Product similarity metric: one metric computes "wide-context" similarity between words using a "bag-of-words" model and a second metric computes "narrow-context" similarity using a bigram language model. The hybrid metric proposed, produces high-quality clustering of words into semantic classes even for semantically heterogeneous domains such as news. The *corpus parser*, re-parses the corpus using the class definitions generated by the *class generator*, i.e., substitutes all instances of each class member with the corresponding class label. The *class generator* and *corpus parser* are run sequentially and iteratively over the corpus.

4.2 Vector Product Similarity in "Bag-of-Words" Model

As was previously mentioned, "Bag-of-words" or Vector Space representation models view each word as one dimension in a high-dimensional vector space. Every document is considered as a vector in this space, with a zero value for every word that does not appear in the document, and a non-zero value for every word that appears in it. The non-zero values are set by using various weighting schemes — for example words that occur frequently in a document are given higher values. Similarity between two documents is measured using the cosine of the (normalized) vectors representing the two documents.

In our work, the Vector Space model is used to calculate the contextual similarity between words that appear in a corpus of documents. The key assumption is that the context surrounding a given word provides information about its meaning.

First, a vocabulary $V = (v_1, v_2, ..., v_N)$ is built containing the N unique words in the corpus. Then, a context window size WS is selected; for each word w in the vocabulary all right and left contexts of length WS are identified in the training corpus, e.g., corpus segments

$$w_{WS,L} \ ... \ w_{2,L} \ w_{1,L} \ w \ w_{1,R} \ w_{2,R} \ ... \ w_{WS,R}$$

where $w_{i,L}$ and $w_{i,R}$ represent the i^{th} word to the left and to the right of w respectively. We define the left-right context set of words $N_{LR,WS}$ as the set of unique words that are found in the left and right contexts of w for a fixed context window size WS.

The feature vector for every word w is defined as $T_{w,WS} = (t_1, t_2, ..., t_N)$ where t_i is a non-negative integer and WS is the context window size. Note that the

feature vector size is equal to the vocabulary size N, i.e., we have a feature for each word in the vocabulary. The i^{th} feature value t_i reflects the occurrences of vocabulary word v_i within the left or right context window WS. These non-negative values, t_i, are set according to one of two different weighting schemes:
Binary Weighting Scheme 'B':

$$t_i^B = \begin{cases} 1, & \text{if} \quad v_i \in N_{LR,WS} \\ 0, & \text{if} \quad v_i \notin N_{LR,WS} \end{cases}$$

Full Weighting Scheme 'F':

$$t_i^F = \begin{cases} c(v_i|WS), & \text{if} \quad v_i \in N_{LR,WS} \\ 0, & \text{if} \quad v_i \notin N_{LR,WS} \end{cases}$$

where $c(v_i|WS)$ is a function of the number of occurrences of v_i within a left and right window context of size WS for the word w and for the whole corpus.

The similarity of two words, w and w', is measured as the cosine distance of their corresponding feature vectors, $T_{w,WS}$ and $T_{w',WS}$. In case of weighting scheme 'B' the similarity, VP_{1B}, is defined as:

$$VP_{1B}(w, w') = \frac{\sum_{i=1}^{N} t_i^B t_i'^B}{\sqrt{\sum_{i=1}^{N}(t_i^B)^2}\sqrt{\sum_{i=1}^{N}(t_i'^B)^2}} \tag{4.2.1}$$

for $T_{w,WS}^B = (t_1^B, t_2^B, ..., t_N^B)$ and $T_{w',WS}^B = (t_1'^B, t_2'^B, ..., t_N'^B)$. In case of weighting scheme 'F', the similarity,VP_{1F}, is calculated using the following equation:

$$VP_{1F}(w, w') = \frac{\sum_{i=1}^{N} t_i^F t_i'^F}{\sqrt{\sum_{i=1}^{N}(t_i^F)^2}\sqrt{\sum_{i=1}^{N}(t_i'^F)^2}} \tag{4.2.2}$$

for $T_{w,WS}^F = (t_1^F, t_2^F, ..., t_N^F)$ and $T_{w',WS}^F = (t_1'^F, t_2'^F, ..., t_N'^F)$.

4.3 Vector Product Similarity in Bigram Language Model

As above, the main idea underlying our approach is that the similarity of context implies similarity of meaning. We assume that words, which are similar in contextual distribution, have a close semantic relation. Consider the following sentences.

A strong *quake* measuring on the Richter Scale...

A strong *earthquake* measuring according to ...

The two italicized words occur in same lexical contexts and they are indeed similar in meaning, referring to the concept of earthquake.

A word, w, is considered with its neighboring words in a sequence

$$...w_{1,L}\ \mathrm{w}\ w_{1,R}....$$

where the words in the left and right contexts are represented by $w_{1,L}$ and $w_{1,R}$ respectively.

The similarity of two words, w_1 and w_2, is estimated as the sum of the left and right context-dependent distances. This sum gives the total "distance" between the probability distributions for, w_1 and w_2 as:

$$D^{LR}(w_1, w_2) = D^L_{12} + D^L_{21} + D^R_{12} + D^R_{21} \tag{4.3.1}$$

where D^L and D^R are the left-context and right-context distance respectively and D_{12} denotes the (possibly asymmetric) distance between w_1 and w_2 [25, 21]. Pargellis et al [21] propose four different distance metrics D, all being computed directly from the conditional probability distributions of contexts given words, e.g., see Eqs. (4.3.3), (4.3.4). In our approach, the Vector Product metric was

used. In order to calculate the semantic distance between two words, we compute the cosine distance between two feature vectors; each feature vector of a word w measures the conditional probability of all possible contexts v_i given that word $p(v_i|w)$, i.e., each vector contains bigram language model probabilities for (context, word) pairs. Given the symmetric nature of the Vector Product metric, Eq. (4.3.1) becomes:

$$VP_2 = VP^{LR}(w_1, w_2) = VP^L_{12} + VP^R_{12} \tag{4.3.2}$$

because $VP^L_{12} \equiv VP^L_{21}$ and $VP^R_{12} \equiv VP^R_{21}$. The two terms of Eq. (4.3.2) are defined as follows [21]:

$$VP^L_{12} == \frac{\sum_{i=1}^{N} p_1^L(v_i|w_1)p_2^L(v_i|w_2)}{\sqrt{\sum_{i=1}^{N} p_1^L(v_i|w_1)^2}\sqrt{\sum_{i=1}^{N} p_2^L(v_i|w_2)^2}} \tag{4.3.3}$$

$$VP^R_{12} == \frac{\sum_{i=1}^{N} p_1^R(v_i|w_1)p_2^R(v_i|w_2)}{\sqrt{\sum_{i=1}^{N} p_1^R(v_i|w_1)^2}\sqrt{\sum_{i=1}^{N} p_2^R(v_i|w_2)^2}} \tag{4.3.4}$$

where $V = (v_1, v_2, ..., v_N)$ is the vocabulary set, and $p_1^L(v_i|w_1)$ is the conditional probability of word v_i preceding w_1 in the corpus given word w_1, i.e., the v_i, w_1 bigram model probability.

4.4 Linear Combination

Combining classifiers [31, 11] is a particularly useful technique for diverse corpora, such as those that involve large amount of noise or unusually high dimensional patterns. A popular and simple way of combining multiple classifiers is simple averaging of the corresponding output values. Weighted averaging has also been proposed, along with different methods of computing the proper classifier weights [28]. In our approach, we have two different forms of the Vector Product metric with the same objective but with different perspective of the input corpus: VP_1

with broad lexical scope and VP_2 that concentrates on the immediate context of each word.

Our goal is to create a combined metric that takes into account both wide-context and narrow-context information. In our experiments, a weighted linear combination of the two metrics was used as follows:

$$VP_C = \lambda_1 VP_1 + \lambda_2 VP_2 \tag{4.4.1}$$

where $\lambda_1+\lambda_2=1$. The algorithm for calculating λ_1 and λ_2 will be presented later in this chapter.

4.5 Grouping Semantically Related Words

The VP_1, VP_2 and VP_C metrics output a list of pairs, ranked according to the semantic similarity of their members, from semantically similar to semantically dissimilar. Words and semantic classes (induced in previous system iterations) are valid members of such pairs. From this list that contains all possible word pairs in the corpus, one has to choose a fixed number of top ranking pairs in order to induce the next set of semantic classes. The mapping from top ranking pairs to classes was achieved using a variant of the algorithm presented in [21]. In their approach, a new class label is created for each pair and the two members are assigned to the new class. However, there is no way to merge more than two lexical units at one step which may lead to a large number of hierarchically nested classes.

Extending the work of Pargellis et al, an algorithm was implemented that creates classes that are allowed to have more than two members. This algorithm examines multiple pairs and finds those pairs that have a common element. Provided that certain conditions are met, a new class label is created and the union

of these pairs is assigned to this class. Assume that the pairs (A,B), (A,C), (B,D) were ranked at the upper part of the list. According to the proposed algorithm, the class (A,B,C,D) will be created. To avoid over-generalizations only pairs that are rank ordered close to each other are allowed to participate in this process. The parameter "Search Margin", SM, defines the maximum distance between two pairs (in the semantic distance rank ordered list) that are allowed to be merged in a single class. Consider the following ranked pairs

Position in List	1	2	3	4	5
Pairs	A B	B C	E F	F G	C D

where A, B, C, D, E, F, G represent candidate words or classes. For $SM = 2$ the classes (A,B,C) and (E,F,G) will be generated, while for $SM = 3$ the classes (A,B,C,D) and (E,F,G) will be generated. By adding the search margin SM constraint it was observed that the semantic homogeny of the created classes was better preserved.

The described algorithm is based only in a simple observation regarding the same member that two pairs may have. The quantity and quality of the resulting classes depends on the size of the list. Generally the pairs, which are near to the top of the list, are composed of lexical units that are strongly related. In contrast, the lower pairs in the list have more poor semantic relationship. The balancing art is to have a large list of pairs in order to derive enough semantic classes, preserving at the same time their semantic homogeny.

4.6 Estimating λ_1 and λ_2

As was previously mentioned, the primary goal is to create a hybrid metric that will take into account both "wide-context" and "narrow-context" information. This was achieved by using a weighted linear combination of the two metrics

VP_1 and VP_2 as follows:

$$VP_C = \lambda_1 VP_1 + \lambda_2 VP_2 \tag{4.6.1}$$

where $\lambda_1+\lambda_2=1$. Parameters λ_1 and λ_2 are calculated using an algorithm that is based on the following key idea: Greater importance should be given to the metric that produces more "qualitative" results. In order to measure the performance of VP_1 and VP_2 the algorithm proposed, measures the average distance between the classes induced by each metric at every iteration for a fixed number of top ranking pairs, N_P. The metric that produces more distinct classes i.e. classes with big distance, should be given greater weight.

Assume for example that VP_1 produces $N_{VP_1}^{N_P}$ classes from the top N_P pairs. These classes are represented as:

$c_i = \{v_{i,1}, v_{i,2}, ..., v_{i,n_i}\} \quad 0 \leq i \leq N_{VP_1}^{N_P}$

where c_i represents the i^{th} class and $\{v_{i,1}, v_{i,2}, ..., v_{i,n_i}\}$ represent the n_i members of the i^{th} class. The distance in general between two classes c_j and c_k is computed as follows:

$$D_{c_j,c_k} = \frac{\sum_{p=1}^{n_j} \sum_{q=1}^{n_k} d(v_{j,p} v_{k,q})}{n_j n_k} \tag{4.6.2}$$

where $d(v_{j,p} v_{k,q})$ represents the semantic similarity between words $v_{j,p}$ and $v_{k,q}$ (calculated in this example by VP_1). Finally, the performance of VP_1 is calculated by averaging over all the possible Ds as follows:

$$D_{avg}^{VP_1} = \frac{\sum_{r=1}^{N_{VP_1}^{N_P}} \sum_{t=r+1}^{N_{VP_1}^{N_P}} D_{c_r,c_t}}{\frac{N_{VP_1}^{N_P}(N_{VP_1}^{N_P}-1)}{2}} \tag{4.6.3}$$

It should be noted that both VP_1 and VP_2 outputs a list of pairs according to their semantic similarity, from semantically similar to semantically dissimilar. In other words, Eq. (4.6.2) and Eq. (4.6.3) represent how close the classes are, so small D means that the classes are distinct and away from each other while bigger

values mean that the classes are close. By combining $D_{avg}^{VP_1}$ and $D_{avg}^{VP_2}$ as show in the following equation and considering that λ_1+λ_2=1, it is easy to calculate the values of λ_1 and λ_2.

$$\frac{\lambda_1}{\lambda_2} = \frac{\frac{1}{D_{avg}^{VP_1}}}{\frac{1}{D_{avg}^{VP_2}}} \tag{4.6.4}$$

These parameters are calculated at every iteration, leading to further improvement of the systems performance.

4.7 Synopsis

In this chapter, our approach for automatic induction of semantic classes was thoroughly presented. The way that the bigram language model and the vector space model are applied is explained in details along with the combination of them. Then, the algorithm for grouping semantically similar words is introduced followed by the algorithm for estimating λ_1 and λ_2.

Chapter 5

Experimental Method

5.1 Corpora

Experiments were made on two different kinds of corpora, an heterogeneous domain generic and a domain specific corpus. The first corpus was the semantically heterogeneous "HR-Net" corpus that was downloaded from the Hellenic Resources Network (http://www.hri.org). Specifically, news in English from the Hellenic Radio (ERA) between 01/01/2005 and 05/11/2005 were gathered, considering every article as a single document. HTML tags were removed from each document. The number of articles in the corpus is 2,082, the total number of words is 549,660, the size of the vocabulary is 22,904 words, the average number of words per document is 264, the maximum number of words found in a document is 1,495 and the minimum 41.

The second corpus we experimented with was the domain specific ATIS corpus. This corpus contains transcribed utterances dealing with travel information. We used an experimental corpus consisting of 1,705 utterances. The total number of words is 19,197 and the size of vocabulary is 575 words.

5.2 Vector Space Model

For the VP_1 metric, as was already mentioned, the feature vector for every word w is defined as $T_{w,WS} = (t_1, t_2, ..., t_N)$ where t_i is a non-negative integer and WS is the context window size. These non-negative values, t_i, are set according to one of two different weighting schemes:

Binary Weighting Scheme 'B':

$$t_i^B = \begin{cases} 1, & \text{if} \quad v_i \in N_{LR,WS} \\ 0, & \text{if} \quad v_i \notin N_{LR,WS} \end{cases}$$

Full Weighting Scheme 'F':

$$t_i^F = \begin{cases} c(v_i|WS), & \text{if} \quad v_i \in N_{LR,WS} \\ 0, & \text{if} \quad v_i \notin N_{LR,WS} \end{cases}$$

where $c(v_i|WS)$ is a function of the number of occurrences of v_i within a left and right window context of size WS for the word w and for the whole corpus.

In the conducted experiments, two versions of t_i^F were used: In the first version,

$$c_1(v_i|WS) = f(v_i|WS) \tag{5.2.1}$$

$f(v_i|WS)$ represents the number of occurrences of v_i (frequency) within a left and right window context for the word w and for the whole corpus.

In the second version,

$$c_2(v_i|WS) = f_1(v_i|WS)log(N_C/N_{v_i}) \tag{5.2.2}$$

$f(v_i|WS)$ represents again the number of occurrences of v_i, N_C represents the total number of documents in the corpus and N_{v_i} the number of documents in which v_i exists at least once.

5.3 Bigram Language Model

For the VP_2 metric, a bigram language model was built using the CMU Statistical Language Modeling toolkit [2], applying Witten-Bell discounting [9]. Since the computation of bigram probabilities in Eq. (4.3.3), (4.3.4) over V is a time consuming procedure for the HR-Net corpus, we focused only on the "seen" bigrams (bigrams that appeared in the corpus), for which no backoff weight is needed. Furthermore, we set a threshold of k "seen" bigrams for each word participating in the pair, in order to reduce computational complexity. We followed this strategy only for the HR-Net corpus by setting $k = 3$, while for the ATIS corpus we did not demand a minimum number of "seen" bigrams to compute the similarity between words.

5.4 Experimental Procedure

The proposed system works iteratively performing the following steps:

Step 1: Calculation of the VP_1 metric.

Step 2: Calculation of the VP_2 metric.

Step 3: Normalization of the VP_1 and VP_2 results using min-max normalization.

Step 4: Calculation of λ_1 and λ_2 using the procedure that was presented in Section 4.6.

Step 5: Calculation of the hybrid VP_C metric.

Step 6: Induction of semantic classes as was presented in Section 4.5.

Step 7: Corpus re-parsing: all occurrences of the derived class members in the corpus are substituted by the corresponding class label.

Step 8: If specified number of iteration SI is reached stop, else go to step 1.

The experimental parameters are:

Parameter 0: The choice of the weighting scheme for the "wide-context" semantic similarity metric VP_1, i.e., use VP_{1F} or VP_{1B} as defined in Sections 4.2 and 5.2.

Parameter 1: The size of the context window WS for metric VP_1 as defined in Section 4.2.

Parameter 2: The number of system iterations (SI).

Parameter 3: The number of induced semantic classes per iteration (IC).

Parameter 4: The size of Search Margin (SM) defined in Section 4.5.

Parameter 5: The number of top ranking pairs (N_P) in order to calculate λ_1 and λ_2 as defined in Section 4.6.

5.5 Synopsis

In this chapter, the experimental procedure that was followed is presented in details. The textual corpora that was used is introduced. Then, every step of the system that was implemented is introduced along with the parameters that we experimented on regarding bigram language model, vector space model and the combination of the two metrics.

Chapter 6

Evaluation

6.1 Evaluation

In order to evaluate the induced semantic classes for the HR-Net corpus, we used as a benchmark a manually crafted semantic taxonomy. Two researchers were assigned this task, devising a taxonomy of 43 semantic classes with 1,820 word-members in them. Every word was assigned to a single class. In order to avoid infrequent words, a threshold was adopted and only words with frequency greater than 9 in the corpus were included in the taxonomy. Regarding the experimental procedure, in order to decrease computation time, our system was tested only for these 1,820 words. The following table illustrates 5 representative handcrafted classes along with example members.

Class Name	Members
Education	university, school, student...
Politics	Karamanlis, president, minister...
Law	prosecutor, judge, crime...
Health	hospital, surgery, pharmaceutical...
Sports	Olympiacos, UEFA, Rehhagel ...

For the evaluation procedure of the ATIS corpus, we used a manually crafted semantic taxonomy, consisting of 32 classes that include a total of 291 members.

Every word was assigned to a single class. Regarding the experimental procedure, we generated manually characteristic chunks, like New York → New_York, J F K → J_F_K etc. Also, during the experiments on ATIS, all the words of the vocabulary were taken under consideration. The following table shows 5 representative handcrafted classes along with some members.

Class Name	Members
City	Atlanta, Dallas, Las_Vegas...
Day	Monday, Tuesday, Friday...
Fairtype	one_way, round_trip, nonstop...
Airline	Delta, Lufthansa, T_W_A...
Meal	meal, lunch, breakfast ...

Although a hierarchical semantic taxonomy was also constructed for both corpora, the evaluation focused only on the flat (terminal) semantic classes presented above. In other words, every induced class was evaluated with respect only to the corresponding handcrafted class without examining its relationships with other classes over the taxonomy. An induced class is assumed to correspond to a handcrafted class, if at least half of its members are included ("correct members") in the handcrafted class. Precision and recall are calculated as follows:

$$\text{Precision} = \frac{\sum_{i=1}^{m} cm_i}{\sum_{i=1}^{m} \alpha_i} \qquad \text{Recall} = \frac{\sum_{i=1}^{m} cm_i}{\sum_{j=1}^{r} \beta_j}$$

where m is the total number of induced classes, r is the total number of handcrafted classes, cm_i is the "correct members" of the i^{th} induced class, α_i is the total number of members of the i^{th} induced class and β_j is the total number of members of the j^{th} handcrafted class that appear in the corpus.

6.2 Evaluation Results on the HR-Net corpus

Regarding HR-Net corpus, many experiments were conducted, each one with different combination of the parameters that were described in Section 5.4. Among these experiments Fig 6.1 illustrates the three metrics, VP_1, VP_2 and VP_C, whose combination lead to the best performance of the system. The parameters used in this experiments were VP_{1F} combined with $c_2(v_i|WS)$ as described in Section 5.2, $WS = 50$, $SI = 30$, $IC = 10$, $SM = 10$, $N_P = 20$. Clearly, the VP_1 metric sig-

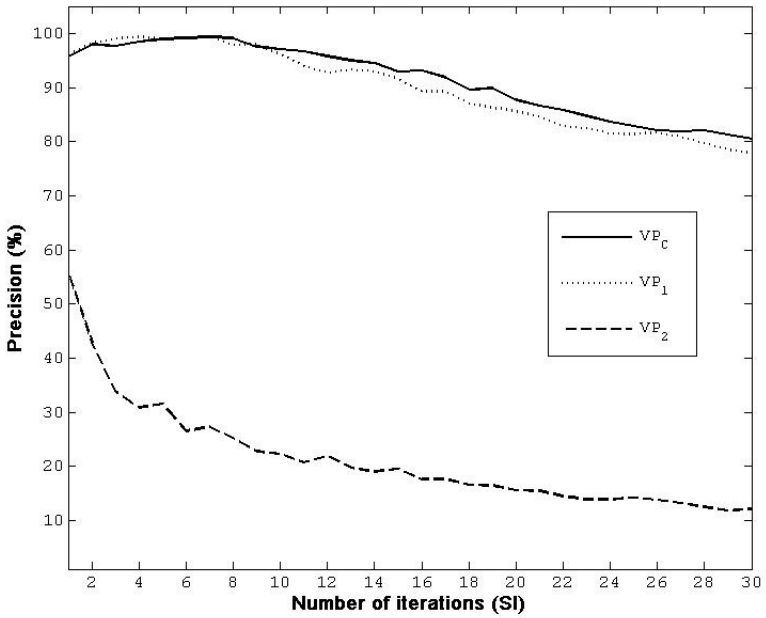

Figure 6.1: Cumulative precision on the HR-Net corpus as a function of system iterations for the three metrics.

nificantly outperforms the VP_2 metric on this task. This is expected because the HR-Net corpus is semantically diverse and words with similar "narrow-context" are often not semantically related. Despite the poor precision of VP_2, this metric contributes to the combined metric VP_C by identifying those words that have a

distinct bigram context. The linear combination of the two metrics, VP_C, tends to achieve the best results but the difference between VP_1 and VP_C is often small or non-existent. The highest precision, 99.31%, is obtained by VP_C at the end of the 7^{th} iteration.

The following table presents the cumulative values of achieved recall as a function of system's iterations and metric used (VP_C, VP_1 or VP_2) on the HR-Net corpus (same parameters as above).

	Recall		
SI	VP_C	VP_1	VP_2
5	5.71%	5.11%	3.30%
10	10.44%	9.50%	5.44%
15	14.28%	13.63%	7.86%
20	17.80%	16.98%	9.78%
25	20.88%	19.89%	12.09%
30	23.74%	22.97%	14.12%

It can be seen, that VP_C has slightly higher recall than VP_1 and that VP_2 is clearly the worst out of the three metrics. Note that at the end this experiment for $SI = 30$ the VP_C metric generates 39 classes with 432 members.

The following figure, fig. 6.2 illustrates the values of λ_1 and λ_2 that were assigned to VP_1 and VP_2 respectively at every system's iteration for the same experiment on the HR-Net corpus. It can be seen that, in general, VP_1 metric produces more qualitative results and that is why λ_1 is assigned bigger values. VP_2 metric contributes to the VP_C metric especially during the first 10 system's iterations where its precision remains above 20% as shown in Fig. 6.1. Nevertheless and despite the poor performance of VP_2 metric, VP_C metric performs better both in terms of precision and recall.

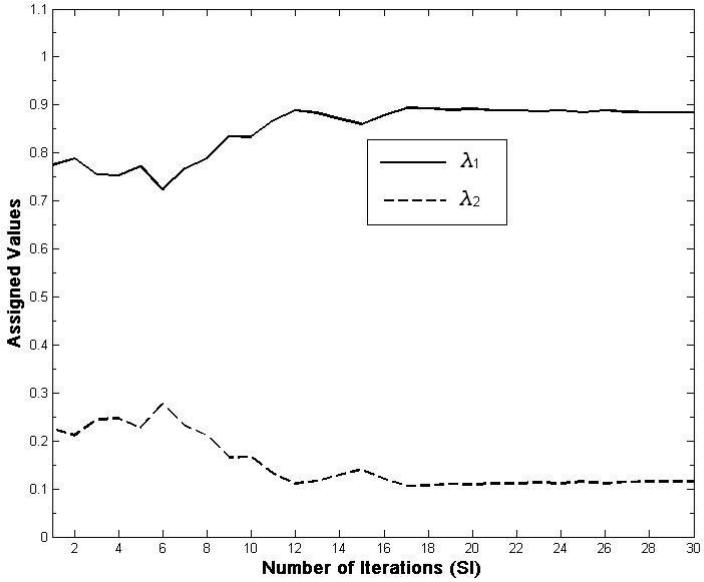

Figure 6.2: Values assigned to λ_1 and λ_2 at each system's iteration.

6.3 Evaluation Results on the ATIS corpus

Among the experiments that were conducted on ATIS corpus, Fig. 6.3 shows the performance of the three metrics VP_1, VP_2 and VP_C, whose combination lead to the best performance of the system. For this experiment we used the following parameters: VP_{1f} combined with $c_2(v_i|WS)$ as described in Section 5.2, $WS = 20$, $SI = 30$, $IC = 10$, $SM = 5$, $N_P = 50$. On this domain-specific corpus the performance of the VP_1 and VP_2 metrics is reversed. VP_2 clearly outperforms VP_1 both in terms of precision and recall (shown in the table that follows). This is expected because in this semantically homogeneous corpus the "narrow-context" similarity often signifies semantic similarity; while there is not enough data to adequately train the statistics of the "wide-context" metric. The precision of the linear combination of the two metrics, VP_C is higher than the

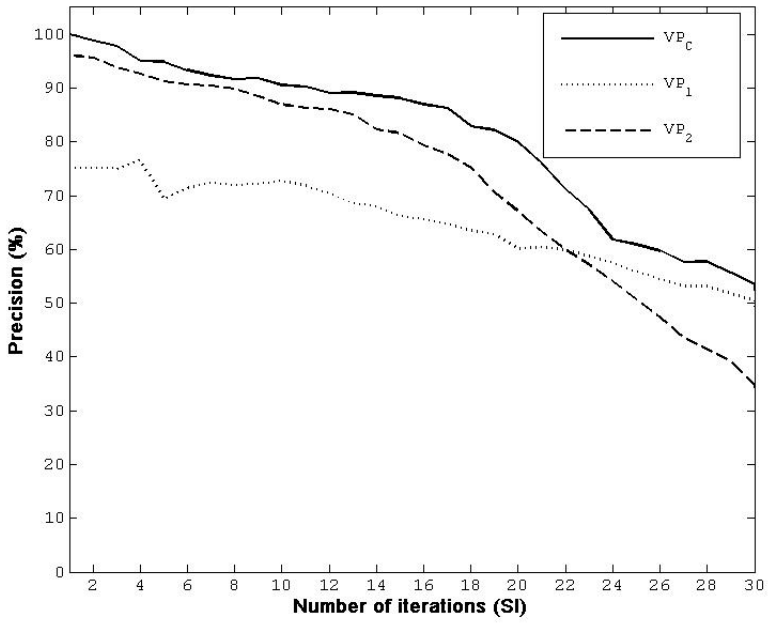

Figure 6.3: Cumulative precision on the ATIS corpus as a function of system iterations for the three metrics.

precision of VP_2, especially after the 14^{th} iteration where the performance of VP_2 drops significantly. Note that the best precision for VP_2 and VP_C (almost 100%) is achieved in the first iteration of the system.

The following table presents the cumulative values of achieved recall as a function of system iterations and metric used.

	Recall		
SI	VP_C	VP_1	VP_2
5	29.55%	8.59%	30.58%
10	40.89%	19.59%	43.30%
15	53.61%	28.52%	53.26%
20	58.42%	39.52%	59.79%
25	65.98%	46.74%	65.29%
30	70.44%	51.89%	71.13%

It can be seen, that VP_C and VP_2 have similar recall values, while VP_1 is significantly worse. Note that for $SI = 30$ the VP_C metric generates 28 classes with 205 members.

The following figure, fig. 6.4 illustrates the values of λ_1 and λ_2 that were assigned to VP_1 and VP_2 respectively at every system's iteration for the same experiment on the ATIS corpus. It can be seen that, in general, VP_2 metric

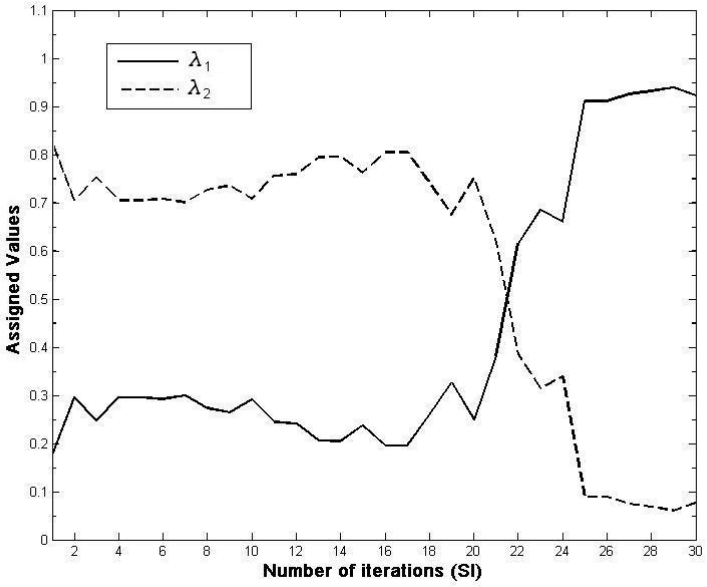

Figure 6.4: Values assigned to λ_1 and λ_2 at each system's iteration.

produces more qualitative results and that is why λ_2 is assigned bigger values especially at the first system's iterations. Later, after the 20^{th} where the performance of VP_2 drops, λ_1 is assigned greater values since VP_1 performs better. Generally, VP_1 metric contributes to the VP_C metric in every system's iteration, leading to better performance both in terms of precision and recall.

Chapter 7

Conclusions

This work raised several valuable issues regarding the nature of human language and how it can be integrated with probability and statistical techniques. It was shown that it is possible to discover the hidden meaning of the natural language without the support of strict grammatical rules. Statistical language processing was used taking into account only the textual context.

An unsupervised procedure for automatic induction of semantic classes was presented using a "wide-context", VP_1, and a "narrow-context", VP_2, semantic similarity metric as well as their combination VP_C. It was shown that VP_1 performs best for the semantically heterogeneous HR-Net corpus, while VP_2 performs better for the domain-specific ATIS corpus. The hybrid metric VP_C performed slightly better than the best of VP_1, VP_2 in our experiments, both in terms of precision and recall. Semantic class precision of 90% and recall of 24% was obtained for the HR-Net corpus. Precision of 88% and recall of 70% was achieved for the ATIS corpus.

We have also presented, an unsupervised procedure to measure the quality of each metric, VP_1 and VP_2, in order to assign values to λ_1 and λ_2. The distance between the classes induced by each metric was calculated, assigning bigger weight to the metric that produced more distinct classes. This measure

proved to perform very well as it lead VP_C to perform better during almost every system's iteration regarding experiments on both corpora.

7.1 Ongoing Work

During the last few weeks, our focus was mainly concentrated on measuring the quality of each metrics' results, in order to calculate λ_1 and λ_2 more efficiently. In Section 4.6 the algorithm presented, takes under consideration only the average distance between classes (inter-class distance). A variant of this algorithm was also applied which considers additionally the average distance between the members of each induced class (intra-class distance) at every system's iteration.

As in Section 4.6, assume that VP_1 produces $N_{VP_1}^{N_P}$ classes from the top N_P pairs. These classes are represented as:

$c_i = \{v_{i,1}, v_{i,2}, ..., v_{i,n_i}\} \quad 0 \leq i \leq N_{VP_1}^{N_P}$

where c_i represents the i^{th} class and $\{v_{i,1}, v_{i,2}, ..., v_{i,n_i}\}$ represent the n_i members of the i^{th} class. The distance in general of a class member $v_{i,l}$ from the other members of the class c_i is computed as follows:

$$d'_{v_{i,l}} = \frac{\sum_{k=1,k\neq l}^{n_i} d(v_{i,l}v_{i,k})}{n_i - 1} \tag{7.1.1}$$

where $d(v_{j,p}v_{k,q})$ represents again the semantic similarity between words $v_{i,l}$ and $v_{i,k}$ (calculated in this example by VP_1). The average distance between the members of the class c_i is:

$$D'_{c_i} = \frac{\sum_{l=1}^{n_i} d'(v_{i,l})}{n_i} \tag{7.1.2}$$

Finally, the average distance between the members of every class at each system's iteration is:

$$D'^{VP_1}_{avg} = \frac{\sum_{i=1}^{N_{VP_1}^{N_P}} D'_{c_i}}{N_{VP_1}^{N_P}} \tag{7.1.3}$$

Eq. (7.1.3) represents how compact the induced classes are. By combining $D_{avg}^{VP_1}$, $D_{avg}^{VP_2}$, $D_{avg}^{\prime VP_1}$ and $D_{avg}^{\prime VP_2}$ as show in the following equation and considering that $\lambda_1+\lambda_2=1$, it is easy to calculate the values of λ_1 and λ_2.

$$\frac{\lambda_1}{\lambda_2} = \frac{\frac{D_{avg}^{\prime VP_1}}{D_{avg}^{VP_1}}}{\frac{D_{avg}^{\prime VP_2}}{D_{avg}^{VP_2}}} \tag{7.1.4}$$

Primary experiments conducted with the ATIS corpus using Eq. (7.1.4) lead to promising results. Fig. 7.1 shows the performance of VP_C (using Eq. (4.6.4)), VP'_C (using Eq. (7.1.4)), VP_1 and VP_2. For this experiment we used the following parameters: VP_{1f} combined with $c_2(v_i|WS)$ as described in Section 5.2, WS = 20, SI = 30, IC = 10, SM = 5, N_P = 50. It is clear that both VP_C and

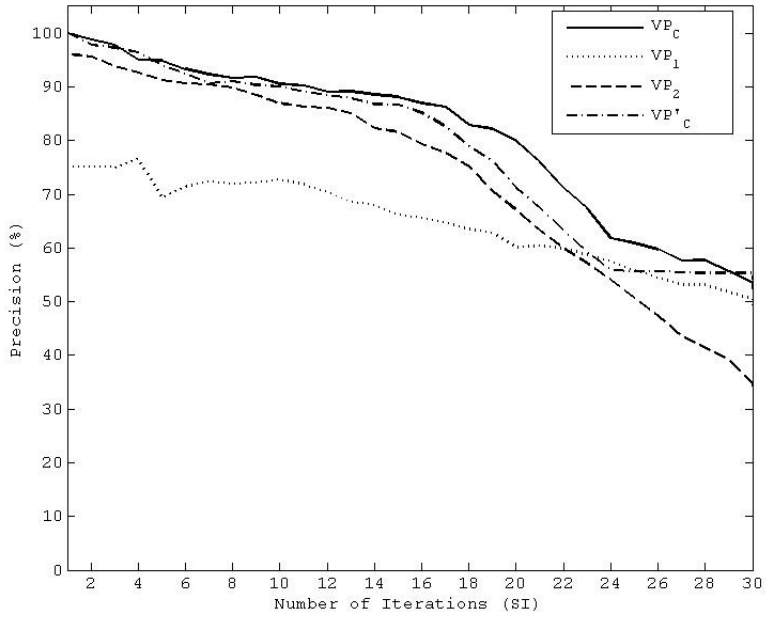

Figure 7.1: Cumulative precision on the ATIS corpus as a function of system iterations for the four metrics.

VP'_C outperforms VP_2 in terms of precision, with VP_C performing slightly better

than VP'_C. Regarding the cumulative values of achieved recall, as shown in the following table, VP'_C performs better than VP_C.

	Recall			
SI	VP'_C	VP_C	VP_1	VP_2
5	30.93%	29.55%	8.59%	30.58%
10	41.24%	40.89%	19.59%	43.30%
15	53.26%	53.61%	28.52%	53.26%
20	61.51%	58.42%	39.52%	59.79%
25	66.67%	65.98%	46.74%	65.29%
30	71.82%	70.44%	51.89%	71.13%

Finally, fig. 7.2 illustrates the values of λ_1 and λ_2 that were assigned to VP_1 and VP_2 respectively at every system's iteration. Again, as in fig. 6.4, VP_2

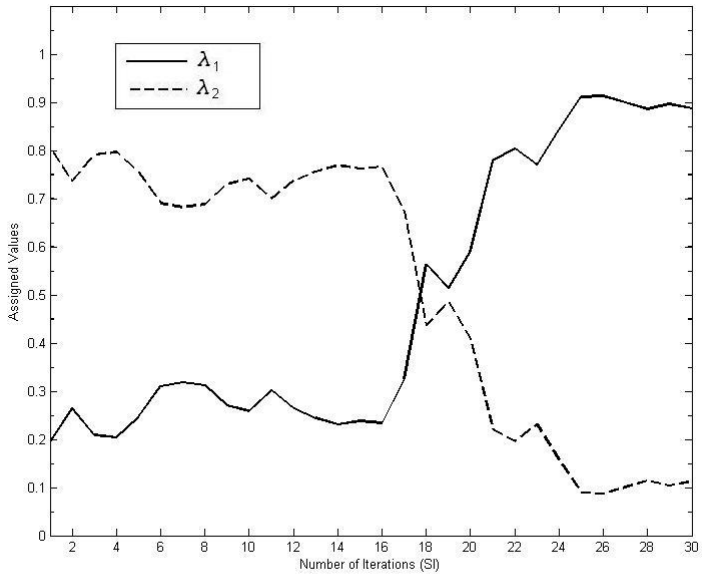

Figure 7.2: Values assigned to λ_1 and λ_2 at each system's iteration.

is assigned bigger weights at the first iterations of the system but during the following iterations the situation is reversed and VP_1 receives greater importance.

7.2 Future Work

More work is needed to improve on each of the metrics and, especially, on the combined metric. Further research needs to be done in the area of soft clustering regarding the non-deterministic assignment of words to classes, resulting to semantically richer taxonomies. Also, a stopping criterion can determine a sufficient number of rules, terminating automatically procedure of the VP_C metric. One challenging scenario is not only to assign names to the output classes, but also to identify various kinds of relations that exist between them, setting the foundations for automatic hierarchy derivation.

During our experimentation we observed that the system generated automatically some internal relationships between previous induced semantic classes. A representative example, taken from the experiments over the HR-Net corpus is shown in Fig. 7.3. This aspect of the system can be considered as a promising

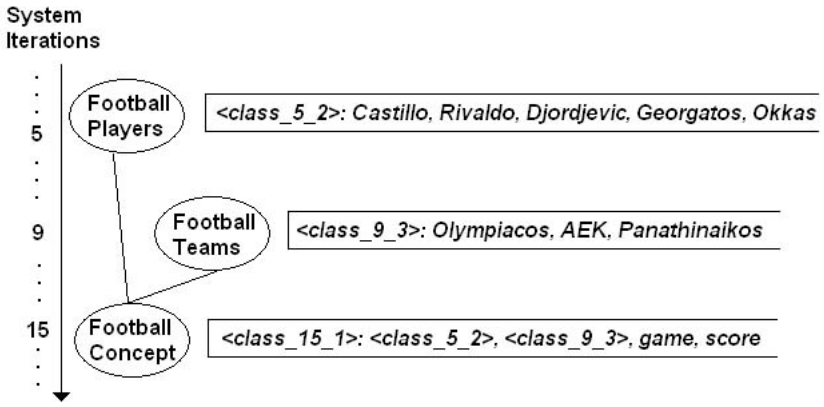

Figure 7.3: Pictorial view of relationships among induced classes for HR-Net

step towards the automatic extraction of a hierarchy between taxonomic classes.

Bibliography

[1] Brown, P. F. et al., "*Class-based n-gram models of natural language.*" Comput. Linguist. 18 (4), 467-479, 1992.

[2] Clarkson, P.R., Rosenfeld, R., "*Statistical Language Modeling Using the CMU-Cambridge Toolkit.*" In: Proc. Fifth European Conf. on Speech Comm. and Tech., 1997.

[3] Cruse, D.A., "*Lexical Semantics.*" Cambridge, U.K.: Cambridge University Press, 1986.

[4] Dagan, I., Lee, L., Pereira, F., "*Similarity-Based Methods for Word-Sense Disambiguation.*" In: Proc. 35th Annual Meeting of the ACL, with EACL 8, 1997.

[5] Duda, R.O., Hart, P.E., Stork, D.G., "*Pattern Classification.*" John Wiley & Sons, Inc, New York, 2001.

[6] Faure, D. and N'Edellec C., "*A Corpus-based Conceptual Clustering Method for Verb Frames and Ontology Acquisition.*" In LREC workshop on Adapting lexical and corpus resources to sublanguages and applications, Granada, Spain, 1998.

[7] Faure, D. and N'Edellec C., "*ASIUM: Learning sub-categorization frames and restrictions of selection.*" 10th Conference on Machine Learning (ECML 98), Workshop on Text Mining, Chemnitz, Germany, 1998.

[8] Francis, W. and Kucera, H., "*The Brown Corpus (Revised and Amplified).*" Brown University, 1979.

[9] Jurafsky, D., Martin, J.H., "*Speech and Language Processing.*" Prentice Hall. Upper Saddle River., 2000.

[10] Katzer, J., McGill, M., Tessier, J., Frakes,W., and DasGupta, P., "*A study of the overlap among document representations.*" Information Technology: Research and Development, 1(4):261274., 1982.

[11] Larkey. S. and Croft, W., "*Combining classifiers in text classification.*" In Proc. SIGIR, 19th ACM International Conference on Research and Development in Information Retrieval, pp. 81-93., 1996.

[12] Lin, D., Pantel, P.,, "*Induction of semantic classes from natural language text.*" In Proceedings of SIGKDD-01. pp. 317322. San Francisco, CA, 2001.

[13] Maddi, G. R. and Velvadapu, C. S., "*Ontology Extraction from text documents by Singular Value Decomposition.*" Bowie State University, 2001.

[14] Maedche, A. and Staab, S., "*Discovering Conceptual Relations from Text.*" Technical Report 399, Institute AIFB, Karlsruhe University, 2000.

[15] Maedche, A. and Staab, S., "*The Text-to-Onto Ontology Learning Environment.*" Institute AIFB, Karlsruhe University, ICCS (International Conference on Conceptual Structures), 2000.

[16] Maedche, A. and Staab, S., "*Ontology learning for the Semantic Web.*" IEEE Intelligent Systems, 16(2), 2001.

[17] Manning, C.D., Schutze, H., "*Foundations of Statistical Natural Language Processing.*" The MIT Press, Cambridge, 2000.

[18] McGill, M., Koll, M., and Noreault, T., "*An evaluation of factors affecting document ranking by information retrieval systems.*" Final report for grant

NSF-IST-78-10454 to the National Science Foundation, Syracuse University., 1979.

[19] Moffat, A., "*Implement the PPM data compression scheme.*" IEEE Transaction on Communications, 38(11):19171921, 1990.

[20] Pangos, A., Iosif, E., Potamianos, A., Fosler-Lussier, E., "*Combining Statistical Similarity Measures for Automatic Induction of Semantic Classes.*" To appear in Proceedings of the 2005 IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop, 2005.

[21] Pargellis, A., Fosler-Lussier, E., Lee, C, Potamianos, A. and Tsai, A., "*Auto-Induced Semantic Classes.*" Speech Communication. 43, 183-203., 2004.

[22] Pargellis, A., Fosler-Lussier, E., Potamianos, A., Lee, C., "*A comparison of four metrics for auto-inducing sematnic classes.*" In: Proc. Automatic Speech Recognition and Understanding Workshop, Madonna di Campiglio, 2001.

[23] Pargellis, A., Fosler-Lussier, E., Potamianos, A., Lee, C., "*Metrics for measuring domain independence of semantic classes.*" In: Proc. 7th European Conf. on Speech Communication and Technology, Aalborg, Denmark, 2001.

[24] Salton G., "*The Smart System: Experiments in Automatic Document Processing.*" Englewood Cliffs: Prentice Hall, NJ, 1971.

[25] Siu, K.-C., Meng, H.M., "*Semi-automatic acquisition of domain-specific semantic structures.*" In: Proc. Sixth European Conf. on Speech Comm. and Tech., Budapest, vol. 5, pp 2039-2042, 1999.

[26] Srikant, R. and Agrawal, R., "*Mining generalized association rules.*" In: Proc. of VLDB 95, pp. 407419., 1995.

[27] Srivastava, S., Lamadrid, J. G. and Karakashyan, Y., "*Document Ontology Extractor.*" CADIP (Center for Architectures for Data-driven Information Processing), 2000.

[28] Tumer, K. and Ghosh, J., "*Theoretical foundations of linear and order statistics combiners for neural pattern classifiers.*" IEEE Trans. Neural Networks, 1995.

[29] Tumer, K. and Ghosh, J., "*Linear and order statistics combiners for pattern classification.*" In Sharkey, A., editor, Combining Artificial Neural Networks, pages 127162. Springer-Verlag., 1999.

[30] Witten, I.H. and Bell, T.C., "*The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression.*" IEEE Transactions on Information Theory, vol. 37(4), 1991.

[31] Woods, K., Kegelmeyer, W. and Bowyer Jr, K., "*Combination of multiple classifiers using local accuracy estimates.*" IEEE Trans. PAMI, 19, 405-410., 1997.